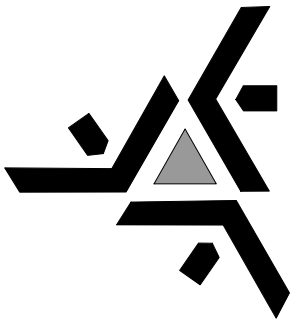

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática
Curso de Engenharia de Produção

Processo de desenvolvimento de um *web site* - Estudo de Caso

Bruno José Pereira da Mota

TCC-EP-34-2009

Maringá - Paraná
Brasil

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática
Curso de Engenharia de Produção

Processo de desenvolvimento de um *web site* – Estudo de Caso

Bruno José Pereira da Mota

TCC-EP-34-2009

Relatório Técnico 4 apresentado como requisito de avaliação no curso de graduação em Engenharia de Produção na Universidade Estadual de Maringá – UEM.
Orientadora: Prof.(ª): Drª Sandra Ferrari

Maringá - Paraná
2009

RESUMO

Para o desenvolvimento de aplicações *web* de qualidade, o uso de uma abordagem sistemática e disciplinada é essencial, dado o crescente aumento do uso e da complexidade de tais aplicações. Nesse cenário, a engenharia de aplicações *web*, ou simplesmente engenharia *web*, torna-se uma atividade mandatória. A engenharia *web* não é uma transcrição exata da engenharia de *software*, pois considera características inerentes às aplicações *web* como, por exemplo, multiplicidade do perfil de usuários e uso de multimídia. A engenharia *web* é apoiada por ferramentas, técnicas e métodos. A escolha desses elementos para um domínio de aplicação pode tornar-se uma tarefa difícil para desenvolvedores e demais interessados e é fundamental para o desenvolvimento efetivo de aplicações *web*. Neste trabalho, é apresentada uma breve história da evolução das páginas eletrônicas, bem como um conjunto de métodos de desenvolvimento de aplicações *web* e suas principais finalidades, em paralelo será desenvolvido um estudo de caso a fim de elucidar as etapas de implementação do processo de desenvolvimento escolhido, culminando no desenvolvimento de um *web site*.

Palavras-chave: engenharia, *web*, métodos de desenvolvimento, estudo de caso.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADV	*Abstract Data View*
ARPANET	*Advanced Research Projects Agency Network*
ASP	*Active Server Pages*
B2B	*Business to Business*
BPM	*Business Process Management*
CERN	Organização Européia para Investigação Nuclear
DARPA	*Defense Advanced Research Projects Agency*
DFD	Diagrama de Fluxos de Dados
DHTML	Dynamic HTML
DNS	*Domain Name System*
E-R	Entidade-Relacionamento
ECO	*Ecosystem of Agile Software Development*
HDM	*Hypermedia Design Model*
HDM-LITE	*Hypermedia Design Model Evolution*
HMBS/M	*Hypertext Model Based on Statecharts/Method*
HTML	*HyperText Markup Language*
HySCHARTS	*Hyperdocument System based on StateCharts*
ISAPI	*Internet Server Application Programming Interface*
JSP	*Java Server Pages*
MER	Modelo Entidade Relacionamento
MVC	Modelo Visão Controle
OO	Orientado a Objetos
OO-H	*Object-Oriented Hypermedia*
OOHDM	*Object Oriented Hypermedia Design Method*

OOWS	*Object-Oriented Web Solution*
OMT	*Object Modeling Technique*
PHP	*Hypertext Preprocessor*
RMDM	*Relationship Management Data Model*
RMM	*Relationship Management Methodology*
RUP	*Rational Unified Process*
SGML	*Standard Generalized Markup Language*
SOA	*Service-Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SWM	*Simple Web Method*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UML	*Unified Modeling Language*
UX	*User Experience*
UWE	UML-*based Web Engineering*
XML	*Extensible Markup Language*
W2000	
W3C	*World Wide Web Consortium*
WAE	*Web Application Extension*
WebML	*Web Modelling Language*
WDSL	*Web Services Description Language*
WWW	*World Wide Web*

1 INTRODUÇÃO

1.1 Contexto

Com a criação do computador, foi possível resolver problemas antes impossíveis ou inviáveis de serem solucionados. O computador funcionava tanto como dispositivo para armazenamento de grandes volumes de dados como solucionador de cálculos gigantescos e complexos, isso tudo através de instruções dadas por humanos. Hoje, muitas tarefas úteis podem ser feitas pelo computador, tais como: gerenciamento da produção de uma indústria, estimativa da melhor rota para a entrega de mercadorias de empresas de logística, controle de transações bancárias entre outros. Pode-se afirmar que atualmente o *software* está presente, explicitamente ou mesmo sem se fazer notar, em todos os aspectos de nossa vida, inclusive nos sistemas críticos que afetam a nossa saúde e bem-estar (PFLEEGER, 2004). Porém, à medida que a capacidade do hardware dos computadores e a complexidade dos problemas aumentavam, também aumentava o tempo para criar o conjunto de instruções ou programas e o número de pessoas necessárias para escrevê-las. Então foram criados métodos sistemáticos para auxiliar o projeto e manutenção desses programas, métodos esses definidos na engenharia de *software*. Por meio da engenharia de *software*, tornou-se viável criar e manusear aplicações com grande número de linhas de código com o mínimo de depuração e envolver vários programadores em um mesmo projeto com a redução de conflitos.

O segundo grande passo da evolução da tecnologia da informação consiste no fato de os computadores do mundo todo compartilharem informações e se comunicarem a grandes distâncias, a Internet. O crescimento da internet causou um impacto significante nos negócios, bancos, indústria, setores de entretenimento, comércio e na vida particular das pessoas (MURUGESAN, 2005). Uma característica que exemplifica essa conseqüência é o tempo de apenas quatro anos que levou para atingir 50 milhões de pessoas (GINIGE, 2000). Originalmente, a internet, ou apenas *web*, foi criada por Tim Berners-Lee como um projeto do CERN (Organização Européia para Investigação Nuclear), para compartilhar informações em grupos de pesquisa (W3C, 2007). No princípio, havia apenas sistemas hipermídia (sistemas compostos por documentos conectados entre si por elos pré-definidos) com pouca ênfase no planejamento de projetos e poucos sistemas foram propriamente testados.

Aplicações *web* estão a cada dia mais presentes e seu desenvolvimento representa uma parte significativa da produção de organizações desenvolvedoras de *software*, bem como de mídia em geral. Embora problemas como atendimento das necessidades de negócio, atraso no projeto, baixa qualidade, entre outros, sejam semelhantes aos encontrados no desenvolvimento de *software* convencional, as soluções não são as mesmas, tornando insuficientes os paradigmas tradicionais da engenharia de *software* e dando origem a uma nova disciplina, a engenharia *web* (KAPPEL et al., 2004).

1.2 Motivação e Relevância

À medida que a dependência do uso das aplicações *web* cresceu, o desempenho, a confiabilidade e a qualidade exigidos para essas aplicações também cresceram. Com isso, a expectativa e a demanda depositadas em tais aplicações aumentaram de modo significante, resultando em um número maior de projetos envolvendo-as.

1.3 Objetivo

Analisar o processo de criação de *web sites*, modelos de desenvolvimento *web*, as etapas produtivas determinadas por tais modelos e escolher qual desses será utilizado para o desenvolvimento do sistema *web* proposto. A opção por uma metodologia de construção do *web site* está baseada exclusivamente na alternativa que mais contribuir para o projeto e implementação do estudo de caso em termos de tempo de concepção.

1.3.1 Objetivo geral

O objetivo genérico da pesquisa é especificar e desenvolver um sistema *web*.

1.3.2 Objetivo específico

Derivando do objetivo geral têm-se os objetivos específicos a serem contemplados à medida que o trabalho se desenrola, tais como: analisar os processos de desenvolvimento *web*, bem como os modelos de construção de *sites* derivados dos processos expostos, adotar o que melhor se adéqüe às necessidades do estudo de caso, elucidar o motivo da escolha, descrever as fases existentes no processo de concepção de *software* tendo como auxilio a elaboração de um *site*, definição das ferramentas e linguagens de programação a serem utilizadas, implementação do aplicativo, testes de qualidade, entre outros.

1.4 Revisão de Literatura

O histórico de redes de computadores teve início na década de 60, quando as redes telefônicas passaram a ceder espaço para redes dedicadas a computadores. A primeira rede foi criada em 1968, pela DARPA (*Defense Advanced Research Projects Agency*), a ARPANET, acrônimo em inglês de *Advanced Research Projects Agency Network* do Departamento de Defesa dos Estados Unidos da América, foi a primeira rede operacional de computadores à base de comutação de pacotes, e a precursora da internet. Nas décadas de 80 e 90, houve um aumento considerável na quantidade de redes, ao mesmo tempo em que houve o desenvolvimento do protocolo TCP/IP e do DNS, oferecendo novas funcionalidades e potenciais para redes, resultando no seu crescimento (TANEBAUM, 1997).

Finalmente, na década de 90, a ARPANET se tornou pública e passou a ser chamada de internet. Com a finalidade de levá-la para fins comerciais, foi criada a WWW (*World Wide Web*) ou mais comumente *web*, introduzindo e popularizando a internet para milhões de pessoas ao redor do mundo.

O W3C (*World Wide Web Consortium*) foi fundado em outubro de 1994 pelo próprio criador da *web* – Tim Berners-Lee, para elevar ao máximo o aproveitamento da *web* por meio do desenvolvimento de protocolos comuns que promoveriam a evolução e a interoperabilidade da mesma (W3C, 2007).

A internet foi a precursora de uma revolução que alterou o comportamento de desenvolvedores de *software* e de seus respectivos usuários. Da mesma forma, a internet foi a base para outra profunda revolução: a forma com que são construídas e que são usadas as aplicações (SHOHOUD, 2003).

Em princípio, a grande rede era usada essencialmente para transmissão de informações, por meio da publicação de páginas simples de conteúdo em documentos no formato hipertexto HTML (*HyperText Markup Language*). À medida que evoluíam o hardware e tecnologias de redes e comunicação, essas páginas se aperfeiçoaram com a adição de imagens, formulários, métodos de envios de dados, até se tornarem complexas aplicações. A internet tornou-se uma verdadeira plataforma para aplicações, propiciando assim condições para o nascimento de um novo conceito na tecnologia da informação: as aplicações *web*.

Nesse ponto é interessante definir o conceito de *software* aplicativo (ou aplicação) como sendo um programa de computador que tem por objetivo o desempenho de tarefas de natureza prática, em geral ligadas a processamento de dados, como o trabalho em escritório ou empresarial.

Para Pressman (2005), uma aplicação *web* compreende todo tipo de aplicação existente para internet, desde um simples *site* até um portal de comércio eletrônico com intenso processamento de informações.

Uma aplicação *web* é desenvolvida com base em uma lógica de negócio, sendo considerada como produto de *software* ou sistema de informática que utiliza uma arquitetura distribuída, pelo menos parcialmente, sob o protocolo HTTP. Como conseqüência, pelo menos parte das interfaces com o usuário é acessível por meio de um navegador (PAULA FILHO, 2003).

A atitude predominante no desenvolvimento de aplicações *web* era: "Vamos fazer rápido, não há tempo para planejar", acabou gerando aplicativos com grande probabilidade de ter problemas como baixo desempenho e falhas. Na *web* problemas como esses são ainda mais graves que no *software* tradicional, pois como afirma Nielsen (2000): a distância entre um *site* e seus concorrentes é sempre de apenas poucos ‘cliques’.

Falta de planejamento, projetos mal feitos e falta de gerenciamento acabam tendo conseqüências sérias. Segundo Ginige e Murugesan (2001), 84% dos sistemas entregues não atendem às necessidades do cliente; 79% dos projetos são entregues com atrasos e 63% têm custo maior que o orçamento previsto. Mais de 50% dos sistemas prontos são de baixa qualidade e faltam funcionalidades necessárias.

Como resultado, desenvolvedores e usuários começaram a se preocupar com a maneira como sistemas *web* complexos estavam sendo criados, bem como com seus níveis de desempenho, qualidade e integridade, surgindo então, a engenharia para a *web*.

Para Ginige e Murugesan (2005), desenvolvimento para a *web* é uma mistura de publicações impressas e desenvolvimento de *software*, entre *marketing* e computação, entre comunicações internas e relações externas, e entre arte e tecnologia.

A engenharia da *web* diz respeito ao estabelecimento e uso de princípios científicos sólidos, de engenharia e de gestão, e abordagens disciplinadas e sistemáticas para o bem-sucedido

desenvolvimento, disposição e manutenção de sistemas e aplicações de alta qualidade baseados na *web* (MURUGESAN e GINIGE, 2005).

A engenharia para *web* não é igual à engenharia de *software* tradicional, mas compartilham muitos conceitos e princípios fundamentais, com ênfase nas mesmas técnicas de gerenciamento e atividades. Há pequenas diferenças na maneira como essas atividades são conduzidas, mas a filosofia que dita uma abordagem disciplinada para o desenvolvimento de um sistema de computador é a mesma (PRESSMAN, 2005).

Diversas ferramentas, notações e métodos surgiram, oferecendo diferentes visões para o processo de desenvolvimento de aplicações *web*, como: OOHDM (SCHWABE et al., 1996), as extensões para a linguagem UML (CONALLEN, 2002), a WebML (CERI et al., 2000a), entre outras.

O HTML é uma linguagem de formatação de documentos, não é uma linguagem de programação, e é composta por *tags* (etiquetas) que permitem descrever a estrutura do documento. A linguagem HTML tem por base a linguagem SGML (*Standard Generalized Markup Language*), a qual é utilizada para descrever a estrutura geral de vários tipos de documentos. A linguagem HTML é um padrão mantido pelo W3C que integra empresas, universidades e institutos.

PHP é a abreviação para *Hypertext Preprocessor* ou em tradução livre "Pré-processador de Hipertexto". É uma linguagem de *script* de código aberto, que tem como objetivo primário a geração de conteúdo dinâmico para páginas da internet. Isso quer dizer que ao invés de criar um programa para gerar e imprimir HTML pode-se escrever HTML com o código PHP embutido para gerar conteúdo dinâmico. Como as *tags* HTML são estáticas, cabe ao PHP ou outra linguagem como ASP (*Active Server Pages*) ou Java a criação de conteúdo dinâmico que existe na *web*. Outra vantagem do PHP é que por ser executado no lado do servidor, seu código-fonte não é exibido ao internauta, que apenas terá acesso ao seu HTML resultante (MELO, 2007).

1.5 Organização do Documento

Neste capítulo foram apresentados: o contexto no qual este trabalho se insere, a motivação, os objetivos genéricos e específicos e a revisão de literatura responsável por definir conceitos chaves associados ao tema do presente trabalho. O restante deste documento está organizado da seguinte forma: no Capítulo 2 são apresentados a definição e os princípios básicos da engenharia de *software*, são expostos também os tipos, as características e a classificação das aplicações *web* e também uma definição e a arquitetura para aplicações *web*, no Capítulo 3 são apresentados um processo genérico de desenvolvimento e métodos de desenvolvimento de aplicações para *web*, no Capítulo 4 é exibido o estudo de caso contemplando todas as fases do processo produtivo genérico, bem como, a utilização de um modelo com o intuito de auxiliar a construção do *web site*, finalmente no Capítulo 5 é apresentada a conclusão, anotando as contribuições e os benefícios provindos do presente trabalho.

2 ENGENHARIA DE *SOFTWARE*

2.1 Considerações Iniciais

Software de computador é o produto que os profissionais de *software* constroem e, depois, mantêm ao longo do tempo. Abrange programas que executam em computadores de qualquer tamanho e arquitetura, conteúdo que é apresentado ao programa a ser executado e documentos, tanto em forma impressa como virtual, que combinam todas as formas de mídia eletrônica (PRESSMAN, 2005).

À medida que a importância do *software* cresceu, a comunidade de *software* tem continuamente tentado desenvolver tecnologias que tornem mais fácil, mais rápido e menos dispendioso construir e manter programas de computador de alta qualidade. Algumas dessas tecnologias são voltadas para um domínio de aplicação especifico, por exemplo, projeto e implementação de *sites* da *web*.

Hoje em dia o *software* assume um duplo papel. Ele é produto e, ao mesmo tempo, veículo para a entrega do produto. Como produto ele disponibiliza o potencial da computação presente no *hardware* do computador ou, mais amplamente, por uma rede de computadores acessível pelo *hardware* local. Quer resida em um telefone celular, quer opere em um computador de grande porte, o *software* é um transformador de informações – produzindo, gerindo, adquirindo, modificando, exibindo ou transmitindo informações que podem ser tão simples como um único *bit* ou tão complexas como uma apresentação multimídia. Como veículo usado para entrega do produto, o *software* age como base para o controle do computador (sistemas operacionais), para a comunicação da informação (redes) e para a criação e o controle de outros programas (ferramentas e ambiente de *software*).

De acordo com (PRESSMAN, 2005), existem sete grandes categorias de *software* de computadores, a saber:

- *Software* de sistemas: é a coleção de programas escritos para servir a outros programas. Alguns - por exemplo: compiladores, editores e utilitários para gestão de arquivos. A área de *software* de sistemas é caracterizada por interação intensa com o *hardware* do computador; uso intenso por muitos usuários; operação concorrente que requer ordenação, compartilhamento de recursos e sofisticada gestão de processo;

- *Software* de aplicação: consiste de programas isolados que resolvem uma necessidade específica do negócio. Aplicações nessa área processam dados comerciais ou técnicos de modo que facilita as operações ou gestão/tomada de decisões técnicas do negócio - por exemplo: processamento de transações no ponto-de-venda, controle de processo de fabricação em tempo real;
- *Software* científico e de engenharia: essas aplicações vão de astronomia à vulcanologia, da análise automotiva de tensões à dinâmica orbital do ônibus espacial, e da biologia molecular à manufatura automatizada;
- *Software* embutido: é aquele que reside dentro de um produto ou sistema e é usado para implementar e controlar características e funções para o usuário final e para o próprio sistema. O *software* embutido pode realizar funções muito limitadas e particulares (por exemplo, o controle de teclado para um forno de microondas);
- *Software* para linhas de produtos: projetado para fornecer uma capacidade específica a ser usada por muitos clientes diferentes, o *software* para linhas de produtos pode focalizar um mercado limitado e especial (por exemplo, produtos de controle de estoque);
- Aplicações da *web*: ou, "*WebApps*", cobrem uma ampla gama de aplicações. Na sua forma mais simples, podem ser pouco mais que um conjunto de arquivos ligados por hipertexto que apresentam informações usando texto e poucos gráficos. No entanto, conforme as aplicações de comércio eletrônico (*e-commerce*) e B2B (*business to business*) crescem em importância, as *WebApps* evoluem para sofisticados ambientes computacionais que fornecem não apenas características isoladas, funções de computação e conteúdo para o usuário final, mas também estão integradas ao banco de dados da empresa e às aplicações do negócio;
- *Software* para inteligência artificial: ou IA, faz uso de algoritmos não numéricos para resolver problemas complexos que não são passíveis de computação ou análise direta. Aplicações nessa área incluem robótica, sistemas especialistas, reconhecimento de padrões (de imagem e de voz), redes neurais artificiais, prova de teoremas e de jogos.

A engenharia de *software* é uma área do conhecimento da computação voltada para a especificação, desenvolvimento e manutenção de sistemas de *software* aplicando tecnologias

e práticas de gerência de projetos e outras disciplinas, objetivando organização, produtividade e qualidade.

Atualmente, essas tecnologias e práticas englobam linguagens de programação, banco de dados, ferramentas, plataformas, bibliotecas, padrões, processos e qualidade de *software*.

Os fundamentos científicos para a engenharia de *software* envolvem o uso de modelos abstratos e precisos, que permitem ao engenheiro especificar, projetar, implementar e manter sistemas de *software*, avaliando e garantindo suas qualidades. Além disso, a engenharia de *software* deve oferecer mecanismos para se planejar e gerenciar o processo de desenvolvimento de um sistema de informação.

Para (PRESSMAN, 2005), a engenharia de *software* é uma estratégia sistemática, disciplinada e quantificável para a programação. Envolve o desenvolvimento, operação e manutenção de *software*, para isso utiliza os seguintes elementos: métodos, ferramentas e processos.

Os princípios básicos da engenharia de *software* são:

- Desenvolvimento de um produto complexo;
- Equipe de pessoas especializadas;
- Aplicação de métodos, técnicas, ferramentas modelos e princípios;
- Planejamento e gerência de custos, prazo e pessoal.

2.2 Aplicações *Web*

Sistemas e aplicações baseadas na *web* (*WebApps*) produzem uma complexa matriz de conteúdo e funcionalidade para uma ampla população de usuários finais. A engenharia da *web* (WebE) é o processo usado para criar *WebApps* de alta qualidade. A WebE não é um clone perfeito da engenharia de *software*, mas toma emprestados muitos dos conceitos e princípios fundamentais da engenharia de *software*. Além disso, o processo WebE enfatiza atividades técnicas e de gestão similares. Há diferenças sutis no modo pelo qual essas atividades são conduzidas, mas a filosofia dominante determina uma abordagem disciplinada para o desenvolvimento de um sistema baseado em computador.

O conhecimento envolvido sobre como desenvolver aplicações *web* intensificou-se nos últimos anos por meio da utilização de engenharia de *software* para construção dessas aplicações (GINIGE, 2002). Apareceram diferentes notações, métodos e ferramentas que oferecem diferentes visões para o processo de desenvolvimento de aplicações *web*, tais como, a WebML (CERI et al., 2000a), o OOHDM (SCHWABE, 1996), as extensões para linguagem UML (CONALLEN, 2002), OOWS (FONS et al., 2003), entre outras. Com diferentes métodos e técnicas para o desenvolvimento de aplicações *web*, a escolha de um método pode tornar-se uma tarefa difícil, levando-se em consideração propósitos comerciais, educacionais ou informativos.

Com o passar dos anos, as aplicações *web* evoluíram e trouxeram consigo diferentes disciplinas como multimídia, ciência da informação e tecnologia de informação e comunicação. Além disso, o aumento do uso dessas aplicações fez com que fossem desenvolvidas de maneira *ad-hoc* em sua maioria, sem um mínimo rigor, abordagem sistemática ou controle e garantia da qualidade (GINIGE, 2002; MURUGESAN e DESHPANDE, 2000; PRESSMAN, 2005). O aumento da complexidade e sofisticação das aplicações *web*, bem como suas características e peculiaridades, trouxeram a necessidade de uma nova abordagem para seu desenvolvimento: a engenharia de aplicações *web* ou apenas engenharia *web*.

O termo aplicação *web* é utilizado para descrever sistemas em ambiente *web*. Para Pressman (2005), uma aplicação *web* pode ser de uma simples página HTML a um *web site* completo.

Os primeiros estudos enfocando a produção de aplicações *web* buscaram diferenciá-la do processo de desenvolvimento de *software* convencional e caracterizar atividades específicas no processo de desenvolvimento, como as apresentadas a seguir (ISAKOWITZ et al., 1995; SCHWABE e ROSSI, 1998; CONALLEN, 2002; CERI et al., 2000b; PRESSMAN, 2005):

- Construção do diagrama de navegação e hipertexto, também chamado de mapa de navegação;
- Construção do modelo de arquitetura da informação;
- Projeto das interfaces, com *layout* de páginas e elementos característicos de HTML.

2.3 Tipos de Aplicações *Web*

Como mencionado anteriormente, desde a invenção da internet até os dias de hoje, as páginas, inicialmente usadas como simples meio de compartilhamento de texto, passaram por vários estágios evolutivos, até se tornarem complexas aplicações, definindo assim categorias de aplicações *web* (MURUGESAN e GINIGE, 2005). As principais categorias de aplicações *web*, em ordem crescente de complexidade, são: páginas informativas, transacionais, de fluxo de trabalho, colaborativas e sociais, apresentadas nas subseções a seguir.

2.3.1 Páginas informativas

As páginas informativas são documentos formatados em linguagem de marcação HTML, utilizadas como meios de informação, não tinham meios de interação com o usuário. Atualmente, são muito usadas para divulgação de notícias, catálogos virtuais, livros *on-line* entre outros.

2.3.2 Páginas interativas

Com a introdução da tecnologia do portal de interface comum ou CGI, que é a forma de comunicação que um servidor *web* emprega para enviar informações entre o navegador e um programa de computador em um servidor *web* (WINMAN, 1997), e mecanismos para elementos dinâmicos dos navegadores, páginas interativas puderam ser criadas. Essas páginas permitem comunicação com o servidor por meio de formulários contendo botões, caixas de seleção e também meios de modificar a página de acordo com entradas do usuário. As principais aplicações para páginas interativas são formulários de registro, jogos *on-line* e personalização da apresentação de informações.

2.3.3 Sistemas transacionais

As páginas transacionais permitem que o usuário não apenas interaja por meio da leitura de informações, mas também permitem modificar informações do servidor via transações. Esse mecanismo foi possível graças à introdução de sistemas de banco de dados nos servidores, que permitem formas de os usuários modificarem e interagirem com dados comuns.

Entre as aplicações comuns para sistemas transacionais estão sistemas de compras, bancários e de reservas de passagens *on-line*.

2.3.4 Sistemas baseados em fluxo de trabalho

O estágio seguinte na evolução da capacidade de aplicações *web*, permitiu que fossem manuseados fluxos de trabalho entre companhias, empresas e autoridades públicas. Isso devido à introdução da tecnologia de *web services*, que permite interoperabilidade entre sistemas, ou seja, computadores e sistemas diferentes podiam se comunicar por intermédio de protocolos comuns.

Exemplos típicos de aplicação são os sistemas B2B, usados para transações comerciais eletrônicas entre parceiros de negócios e de administração pública, dentre outros.

2.3.5 Sistemas colaborativos

Sistemas colaborativos têm como função oferecer um ambiente comum de trabalho e métodos de coordenação das interações dos usuários de um grupo. Por meio de um sistema colaborativo, as pessoas podiam trabalhar em documentos comuns e participar de reuniões virtuais mediadas pelo sistema. Exemplos desses sistemas são as salas de bate-papo e os sistemas *wiki*, usados para gerenciamento colaborativo de informações.

2.3.6 Sistemas sociais

Em princípio a internet tinha como característica o anonimato dos usuários, mas havia uma tendência de se ter uma rede onde pessoas se conhecessem. Surgiram então os sistemas sociais, onde pessoas registram informações pessoais, conhecem pessoas e podem fazer buscas de indivíduos com perfil desejado. Entre esses sistemas, estão os *weblogs*, para registro e gerenciamento de informações pessoais e páginas de cadastro pessoal compartilhado.

2.4 Atributos de Sistemas e Aplicações Baseados na *Web*

No inicio da *World Wide Web*, entre 1990 e 1995, os "*sites web*" eram formados de pouco mais do que um conjunto de arquivos de hipertexto ligados, que apresentavam informação usando texto e um pouco de gráficos. Com o passar do tempo, a HTML foi crescendo com ferramentas de desenvolvimento (por exemplo, XML, Java) que habilitaram os engenheiros *web* a fornecer capacidade computacional junto com informação. Sistemas e aplicações baseados na *web* nasceram. Hoje em dia, *WebApps* evoluíram para ferramentas

computacionais sofisticadas que não fornecem somente funções isoladas para o usuário final mas também são integradas com banco de dados corporativos e aplicações de negócio.

Powell et al. (1998) resume as principais diferenças, quando afirma que sistemas baseados na *web* "envolvem uma mistura de publicação impressa e desenvolvimento de *software*, de comercialização e computação, de comunicações internas e relações externas, e de arte e tecnologia". Os seguintes atributos são encontrados na maioria das *WebApps*:

- Concentração em redes: uma *WebApp* reside em uma rede e precisa servir às necessidades de uma comunidade diversificada de clientes. Uma *WebApp* pode residir na internet (consequentemente, permitindo comunicação aberta ao mundo todo). Alternativamente, uma aplicação pode ser colocada em uma intranet (implementando as comunicações dentro de uma organização) ou em uma extranet (comunicação entre redes);
- Concorrência: um grande número de usuários pode ter acesso às *WebApps* ao mesmo tempo. Em muitos casos, os padrões de utilização entre os usuários finais vão variar muito;
- Carga imprevisível: o número de usuários das *WebApps* pode variar por ordens de magnitude de um dia para outro. Cem usuários podem aparecer na segunda-feira; dez mil podem usar o sistema na terça-feira;
- Desempenho: embora a expectativa de 100% de disponibilidade seja irracional, usuários de *WebApps* populares frequentemente querem acesso na base de "24/7/365" (vinte e quatro horas por dia, sete dias na semana e trezentos e sessenta e cinco dia no ano). Usuários da Austrália ou da Ásia podem querer acesso durante horários em que as aplicações domésticas de *software* tradicional da América do Norte poderiam estar fora do ar para manutenção;
- Voltada a dados: a função principal de muitas *WebApps* é usar hipermídia para apresentar conteúdos de texto, gráficos, áudio e vídeo ao usuário final. Além disso, *WebApps* são comumente usadas para dar acesso à informação que existe em bancos de dados que não eram originalmente parte integral de um ambiente baseado na *web* (por exemplo, comércio eletrônico – *e-commerce* – ou aplicações financeiras);

- Sensível ao conteúdo: a qualidade e natureza estética do conteúdo permanecem como um considerável determinante da qualidade de uma *WebApp*;
- Evolução continuada: diferente do *software* de aplicação convencional, que evolui a uma série de versões planejadas e cronologicamente espaçadas, as aplicações *web* evoluem continuamente. Não é incomum que algumas *WebApps* (especialmente, seu conteúdo) sejam atualizadas por meio de um cronograma minuto a minuto ou que o conteúdo seja independentemente calculado para cada solicitação;
- Imediatismo: a necessidade premente de obter *software* para ser colocado rapidamente no mercado – uma característica de muitos domínios de aplicação, as *WebApps* frequentemente exigem um prazo de colocação no mercado que pode ser questão de uns dias ou semanas. Os engenheiros *web* precisam usar métodos de planejamento, análise, projeto, implementação e teste que tenham sido adaptados aos cronogramas de tempo reduzido, requeridos para o desenvolvimento de *WebApp*;
- Segurança: como as *WebApps* estão disponíveis por meio de acesso em rede, é difícil, senão impossível, limitar a população de usuários finais que podem ter acesso à aplicação. A fim de proteger conteúdo reservado e fornecer modos seguros de transmissão de dados, fortes medidas de segurança precisam ser implementadas em toda a infra-estrutura que apóia uma *WebApp* e na aplicação propriamente dita;
- Estética: uma inegável parte da atração de uma *WebApp* é o seu aspecto. Quando uma aplicação é projetada para o mercado ou para vender produtos ou idéias, a estética pode ter tanto a ver com o sucesso quanto o projeto técnico.

Conforme pode ser observado na Tabela 2.1 existem várias classificações de aplicações baseadas na *web*.

Procurando resumir as características encontradas nas *WebApps*, Domingues (2005), anota as classificações de *WebApps* desde as amplas (PRESSMAN, 2005) até as mais elaboradas (POWELL et al., 1998).

Tabela 2.1: Relação de inclusão entre classificações de Aplicações Baseadas na *web* (DOMINGUES, 2005).

<table>
<tr><td>Classificação</td><td colspan="5">Categorias</td></tr>
<tr><td>Powell (1998)</td><td>sites web estáticos</td><td>sites web estáticos com formulários de entrada de dados</td><td>sites web com acesso a dados dinâmico</td><td>sites web criados dinamicamente</td><td>aplicações web</td></tr>
<tr><td>Araújo (2001)</td><td colspan="2">páginas de informações</td><td colspan="2">aplicações web centradas em bancos de dados</td><td>aplicações web que apóiam a realização de transação de negócios</td></tr>
<tr><td>Conallen (2002)</td><td colspan="2">sites web</td><td colspan="3">aplicações web</td></tr>
<tr><td>Pressman (2005)</td><td colspan="5">aplicações web</td></tr>
</table>

Além da classificação apresentada na Tabela 2.1, nota-se a existência de outro tipo de aplicação dentro do ambiente *web*: os *web services*, os quais são componentes de *software* distribuídos pela internet e que utilizam tecnologias abertas como XML, SOAP, WDSL (W3C, 2007) que apóiam o desenvolvimento rápido e de baixo custo de aplicações (PAPAZOGLOU e GEORGAKOPOULOS, 2003). Esses serviços podem ser implementados, oferecidos e utilizados por empresas, provendo uma infra-estrutura para colaboração e integração entre aplicações. Desse modo, estão relacionados ao paradigma de computação orientada a serviços, que utilizam os *web services* como elementos essenciais para a construção de aplicações, envolvendo camadas de serviços, funcionalidades e regras descritas na chamada arquitetura orientada a serviços SOA (*Service-Oriented Architecture*).

Geralmente, para usuários que utilizam uma aplicação *web*, os *web services* são transparentes, uma vez que são componentes de *software*, mas ao invés de serem utilizados localmente, o são de maneira distribuída, por meio do ambiente *web*.

Para este trabalho a definição de aplicação *web* é: aplicações executadas no ambiente *web*, envolvendo aspectos hipermídia, com hipertextos e multimídia, combinados com a lógica de aplicações convencionais.

2.5 Classificação das aplicações *web* sem consenso na comunidade científica

Segundo (GINIGE, 2002) as aplicações web podem ser categorizadas como:

Funcional: oferecer serviços. Exemplos: bancos, comércio eletrônico, bibliotecas digitais.

Informativo: oferecer informação. Exemplos: indicadores on-line, jornais e revistas, rádios.

Entretenimento: oferecer diversão. Exemplos: museus, livros, relacionamento pessoal, sexo.

A junção dos três tipos, funcional, informativo e entretenimento nomeia-se portal. A Figura 2.1 ilustra as três grandes áreas em que se dividem as aplicações *web*, bem como exemplifica as *WebApps* de cada grande grupo.

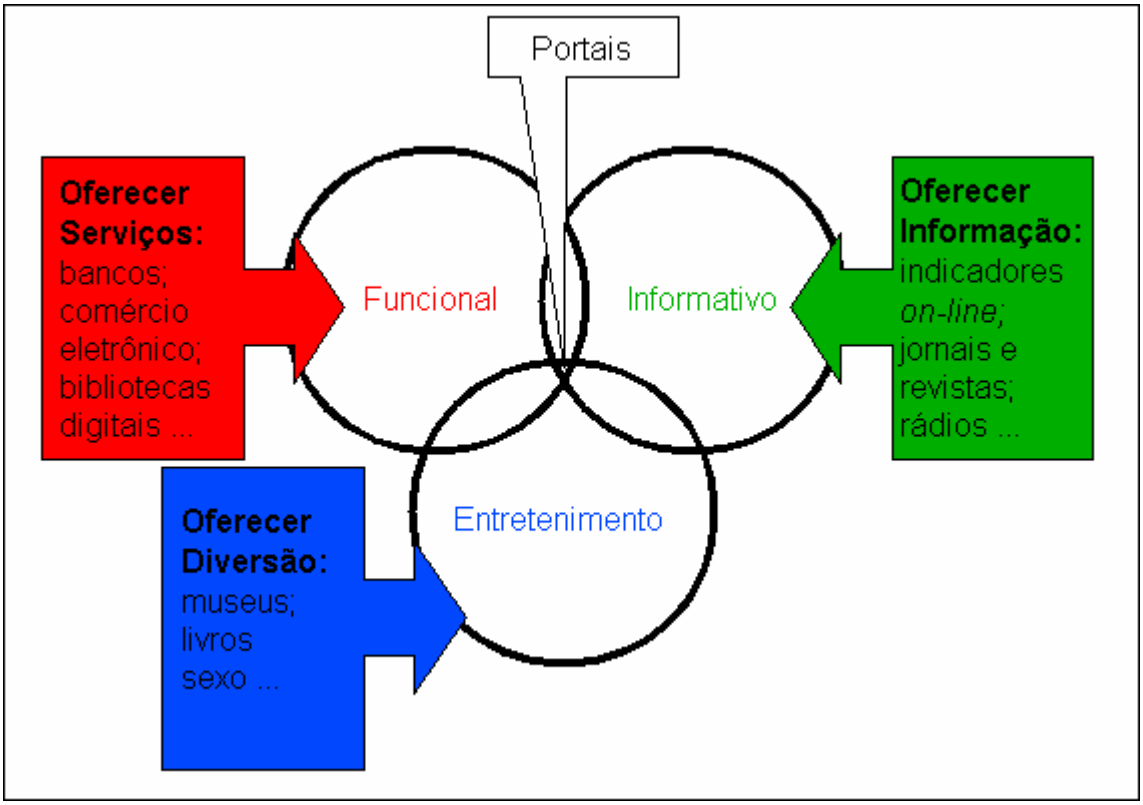

Figura 2.1: Classificação genérica das aplicações *web* (GINIGE).

2.6 Arquitetura

Para o desenvolvimento bem-sucedido de aplicações *web*, o conhecimento de sua arquitetura é um requisito fundamental. De um modo geral, as aplicações *web* possuem três componentes básicos: um servidor *web*, uma conexão de rede e um cliente.

O servidor *web* é um *software* executado em um computador remoto que responde a solicitações de outro *software* chamado cliente, via uma conexão de rede previamente estabelecida. Esses componentes podem ser estruturados em uma arquitetura composta por três camadas. Essas camadas separam as funções de interface com o usuário (apresentação), as funções de acesso ao banco de dados e as funções de lógica de negócio existentes em uma aplicação *web*. Por meio de um servidor *web* pode-se ter acesso às funções de lógica de negócio e de acesso aos dados, enquanto que o cliente possui funções de interface com o usuário (CONALLEN, 2002). Na Figura 2.2 pode-se observar as três camadas e as tecnologias relacionadas.

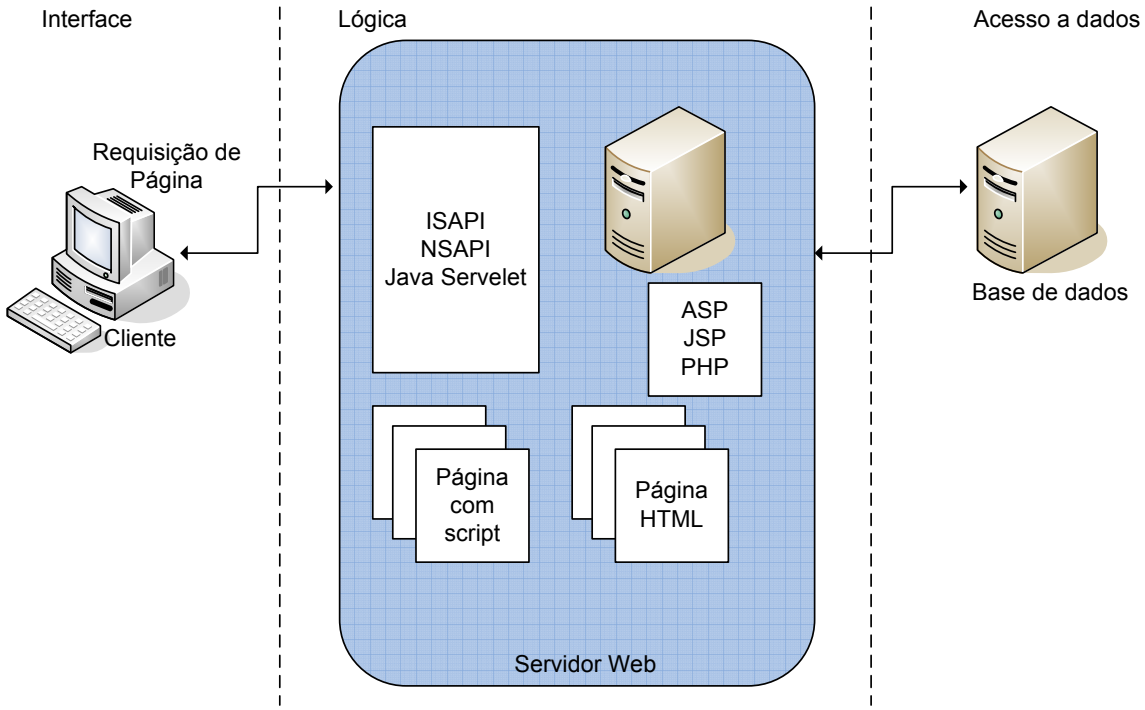

Figura 2.2: Arquitetura de aplicações web (CONALLEN, 2002).

Um exemplo de estilo de arquitetura padrão em três camadas, utilizada no desenvolvimento de aplicações *web* é o MVC (*Model-View-Controller*), o qual separa a funcionalidade envolvida no processamento e apresentação dos dados da aplicação e a lógica de negócio que

regem o acesso aos dados. O modelo (*Model*) mantém o estado persistente dos dados e fornece ao controlador (*Controller*) a capacidade de ter o acesso às funcionalidades da aplicação, encapsuladas pelo próprio modelo. O componente de visualização (*View*) apresenta o conteúdo de um modelo e coleta os dados para o modelo; e especifica como esses dados devem ser apresentados e encaminhados para o controlador das ações do usuário. O controlador é o componente que define o comportamento da aplicação, ou seja, ele interpreta as ações do modelo que, por exemplo, ativam processos da lógica de negócio ao alterar o estado do modelo. O controlador se encarrega de selecionar qual visualização será mostrada a partir da interação com o usuário (SUN MICROSYSTEMS, 2009).

3 ENGENHARIA *WEB*

A engenharia *web* é a aplicação de abordagens sistemáticas, disciplinadas e quantificáveis para o desenvolvimento, operação e manutenção de aplicações baseadas na *web* (DESHPANDE et al., 2003). A adaptação de métodos e técnicas, em função das características das aplicações *web*, torna a engenharia *web* um campo multidisciplinar que incorpora entradas de diversas áreas da engenharia de *software*, bem como as áreas de interação usuário-computador, gerenciamento de projetos, multimídia, hipermídia e projeto gráfico (MURUGESAN e DESHPANDE, 2000).

De um modo geral os elementos que compreendem a engenharia *web* acabam mesclando atividades e aspectos encontrados na engenharia de *software* com atividades e aspectos únicos. Os elementos que podem ser listados para a engenharia *web* são (MURUGESAN e GINIGE, 2005):

- Especificação e análise de requisitos;
- Desenvolvimentos de métodos e técnicas para as aplicações *web*;
- Integração com sistemas legados;
- Migração de sistemas legados para o ambiente *web*;
- Desenvolvimento de aplicações *web* de tempo real;
- Validação, verificação e testes de aplicações *web*;
- Garantia e controle da qualidade;
- Gerência de configuração e de projeto;
- Métricas *web* – métricas para estimativas dos esforços de desenvolvimento;
- Especificação e avaliação de desempenho;
- Atualização e manutenção;
- Desenvolvimentos de equipes e modelos;
- Aspectos humanos e culturais;
- Desenvolvimento centrado no usuário, envolvimento e verificação de *feedback* do usuário;

- Desenvolvimento de aplicação de uso facilitado ao usuário; e
- Treinamento e educação.

Atividades como integração e migração de sistemas legados, elementos como métricas *web* e aspectos humanos e culturais surgem para a engenharia *web*, criando novos desafios para os desenvolvedores e profissionais da área. Além disso, algumas atividades relacionadas, também utilizadas pela engenharia de *software*, apresentam mudanças devido às características das aplicações *web* (Seção 2.4), diferenciando a engenharia de *software* tradicional da engenharia *web*. Por exemplo, as atividades de Validação, Verificação e Teste devem ter suas técnicas adaptadas para que abordem características como o ambiente *web* e questões de navegabilidade.

3.1 Processo de Desenvolvimento

O processo de desenvolvimento é o conjunto de fases e atividades necessárias ao desenvolvimento de um *web site*. Para que os elementos listados na seção anterior possam ter seus objetivos cumpridos, os mesmos devem ser executados de acordo com um processo de desenvolvimento. Pressman (2005) sugere um processo evolutivo e incremental, similar ao modelo de ciclo de vida espiral da engenharia de *software* tradicional. O processo inicia-se com a atividade de formulação, que consiste em identificar as metas e objetivos da aplicação *web* e estabelece o escopo inicial. Na atividade seguinte, o planejamento, é estimado o custo global do projeto e são avaliados riscos associados com o esforço de desenvolvimento e ainda são definidos cronogramas para as fases iniciais e fases subsequentes. Na análise são estabelecidos requisitos técnicos de uma aplicação *web* e são identificados os itens de conteúdo que serão incorporados, além de requisitos de estética. A atividade de engenharia incorpora duas tarefas paralelas, uma consiste no projeto, produção de conteúdo de texto, gráficos, áudio e vídeo, e outra tarefa consiste no projeto técnico. Na atividade de geração de páginas faz-se uso de ferramentas para a criação de aplicações *web*, combinando os projetos de interface, de arquitetura e de navegação para produzir páginas executáveis que possam ser testadas na atividade de teste. Por fim, na atividade de avaliação pelo cliente, podem ser solicitadas modificações que incidem no escopo do projeto e que são integradas em uma futura iteração, por meio do fluxo de processo evolutivo. Na Figura 2.3 é possível observar o modelo de processo proposto por Pressman (2005).

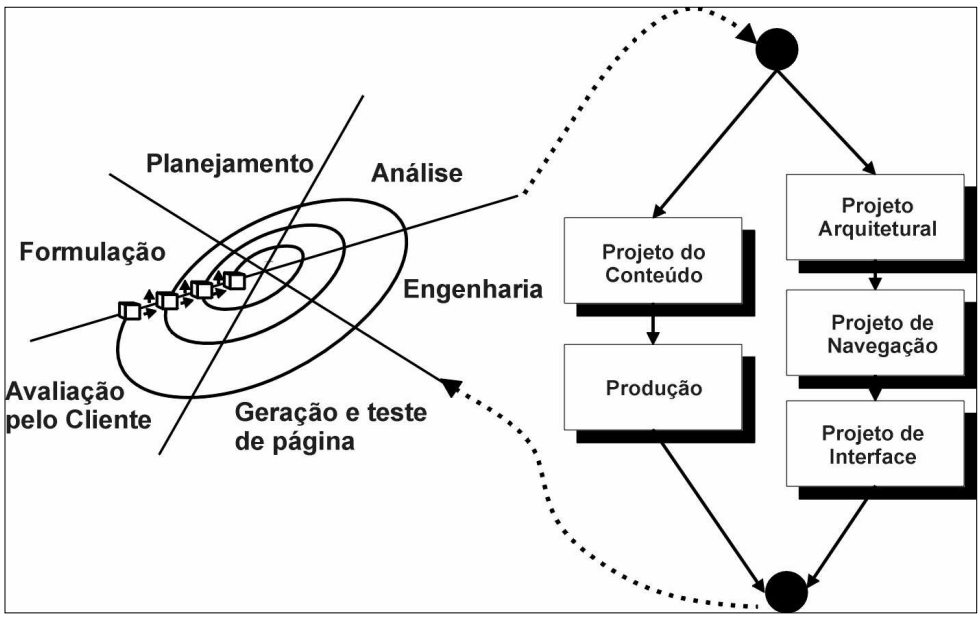

Figura 3.1: Modelo de processo de desenvolvimento de aplicações web proposto por Pressman (2005).

Outro processo que procura considerar os aspectos particulares das aplicações *web* é o processo proposto por Ginige e Murugesan (2005). Assim como o processo anteriormente apresentado, esse processo também é considerado evolutivo, dado que essa característica de processo é útil para a compreensão do contexto no qual a aplicação será desenvolvida, facilita a elicitação de requisitos, ajuda na integração entre diferentes disciplinas e auxilia na comunicação entre diferentes membros. Além disso, também apóia a evolução continuada, facilita o gerenciamento de conteúdo e, de um modo geral, ajuda a controlar a complexidade e diversidade inerente ao processo de desenvolvimento das aplicações *web* (GINIGE e MURUGESAN, 2001).

No processo proposto por Ginige e Murugesan (2005), o primeiro passo, e mais importante, é a análise de contexto, na qual os objetivos, os requisitos e as necessidades dos usuários e da organização que precisa da aplicação são levantados. Nesse passo, a percepção de que os requisitos podem mudar e evoluir, é uma questão chave para o sucesso do processo. No projeto da arquitetura do sistema devem ser considerados os requisitos funcionais, não funcionais e técnicos, além de atributos de qualidade como escalabilidade, manutenibilidade e requisitos de desempenho, pois pode ser difícil ou impossível adicionar esses elementos em fases posteriores do processo.

Em conjunto com o projeto da arquitetura do sistema, um modelo conceitual do projeto deve ser construído para elucidar os principais requisitos e objetivos do sistema. Após as fases de levantamento e análise de requisitos, segue a construção de um plano de projetos, para iniciar o desenvolvimento da aplicação. Após o desenvolvimento há a entrega da aplicação, e por fim, a aplicação mantém-se sob constante avaliação e manutenção. Aliado a essas fases, somam-se atividades de apoio como gerenciamento de projeto, garantia e controle de qualidade e documentação. Na Figura 3.2 pode-se ter uma visão geral desse modelo de processo e verificar como suas atividades estão conectadas.

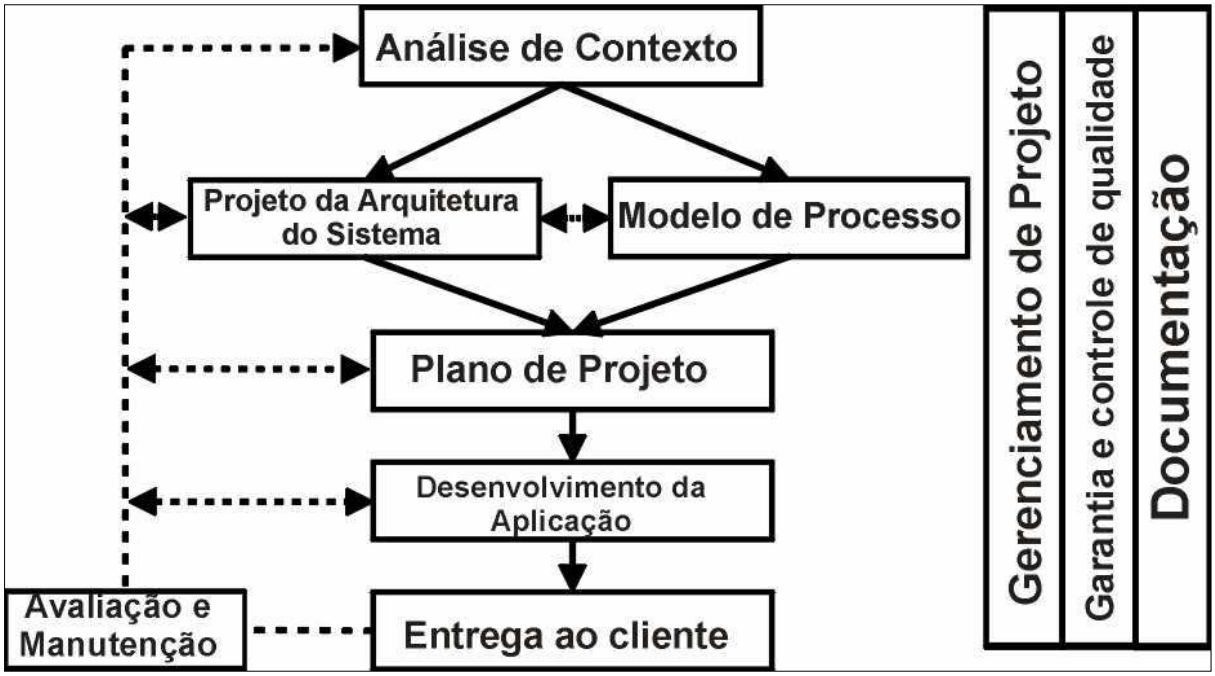

Figura 3.2: Modelo de desenvolvimento de aplicações *web* proposto por Ginige e Murugesan (2005).

As etapas abordadas por esses processos dão ênfase às fases iniciais, de um modo geral a análise de requisitos e o planejamento do projeto. Desse modo, procuram minimizar o possível impacto de futuras alterações e de novos requisitos. Além dessa semelhança, há também o fato de serem processos evolutivos, que procuram adequar-se à característica evolutiva das aplicações *web*. Outras características das aplicações *web* (Seção 2.4) também são consideradas entre as etapas dos processos, com destaque para a fase de engenharia apresentada por Pressman (2005) e a fase do projeto de arquitetura do sistema de Ginige e Murugesan (2005). Por exemplo, na fase de engenharia proposta por Pressman (2005), há uma atividade para cuidar do conteúdo da aplicação *web* (Projeto de Conteúdo), assim como, organizar os elementos visuais (Projeto de Navegação e de Interface). Em relação ao tipo do

processo, no processo proposto por Ginige e Murugesan (2005), as etapas de projeto não são organizadas em um modelo espiral como o proposto por Pressman (2005), mas apresentam-se em um modelo mais próximo ao cascata, permitindo que etapas já realizadas sejam revisitadas, tornando-o iterativo.

Contudo, salienta-se que os dois modelos de processos propostos não especificam nenhuma técnica de modelagem ou método para sua utilização.

A seguir são expostos alguns motivos para se ter um modelo de processo de desenvolvimento, quais sejam:

- Tipicamente um *web site* é desenvolvido de maneira *ad-hoc* - implementar e testar;
- As pessoas consideram que basta saber HTML, CGI e um pouco de programação, para desenvolver um *site*;
- Normalmente não são feitas a definição de objetivos e requisitos, o design, os testes e a manutenção do *web site*.

Cada um dos modelos anteriormente expostos tem suas próprias características, particularidades e ordem para cada uma das etapas. Porém, independente do modelo a ser seguido, a implementação de engenharia para *web* deve contemplar as seguintes atividades:

- Definição do problema;
- Análise e especificação de requisitos;
- *Design* (estrutura organizacional, navegacional, conteúdo, interface da página, funcional);
- Implementação;
- Testes;
- Instalação;
- Evolução e manutenção.

Na seqüência do trabalho é detalhada cada uma dessas etapas citadas acima.

3.1.1 Definição do problema

É nessa atividade que cliente e desenvolvedor estabelecem as metas e os objetivos para a construção da aplicação *web*.

São levantados os propósitos e a motivação para o *site* como, por exemplo, aumentar as vendas, aprimorar o atendimento ao cliente, oferecer novos serviços e acessar as informações. Também são definidos os propósitos da aplicação e o público ao qual se destina. Para tanto, é necessário desenvolver um perfil dos usuários. Esse perfil deve abranger as características relevantes relacionadas aos usuários em potencial, como por exemplo, seus conhecimentos e preferências.

Ainda na fase inicial, é importante que, apontadas as possíveis soluções para que se atinjam os objetivos traçados para a aplicação, seja realizado um estudo de viabilidade das mesmas.

3.1.2 Análise e especificação de requisitos

Na análise, são estabelecidos os requisitos técnicos da aplicação *web*. São identificados os requisitos de dados, funcionais e comportamentais, a fim de criar um modelo completo de análise para a aplicação.

Pressman (2005) subdivide a análise para a engenharia web em quatro tipos:

- Análise de conteúdo: é identificado todo o conteúdo que será fornecido pela aplicação, esse conteúdo pode ser disponibilizado através de textos, gráficos, imagens, vídeo ou áudio;
- Análise de interação: é descrito o modo como o usuário irá interagir com a aplicação *web*;
- Análise funcional: são descritas todas as operações e funções da aplicação;
- Análise de configuração: devem ser detalhados o ambiente e a infra-estrutura na qual a aplicação irá residir, podendo ser na internet, intranet ou em uma extranet.

A evolução contínua das aplicações *web* torna obsoleta a documentação desenvolvida na fase de análise. Esse fato faz com que os desenvolvedores simplesmente não façam a especificação

dos requisitos. Porém, é necessária a definição de um modelo de análise como base para a atividade de *design*.

3.1.3 *Design*

A atividade de *design*, também conhecida como projeto, abrange os aspectos relacionados à estrutura global da aplicação, navegação e interface.

3.1.3.1 Estrutura global da aplicação

De acordo com Pressman (2005), a definição da estrutura da aplicação para *web* deve buscar atingir as metas estabelecidas, levar em consideração o conteúdo que será apresentado e os usuários que irão utilizá-las.

A seguir são apresentadas três diferentes estruturas para *WebApps*:

3.1.3.1.1 Estrutura linear

Utilizada quando há seqüência previsível de interações; um exemplo seriam apresentações de tutoriais com várias páginas de informação, sendo que, nesse caso, o conteúdo é predominantemente linear, a Figura 3.3 apresenta o esquema de como funciona a estrutura linear.

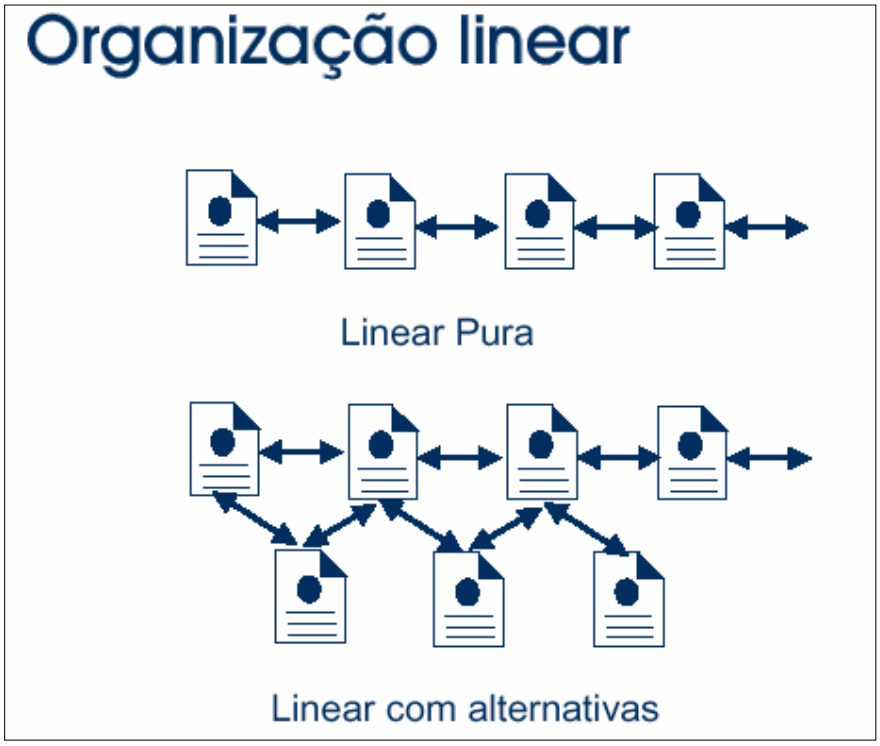

Figura 3.3: Esquema de organização linear de *WebApps*.

3.1.3.1.2 Estrutura hierárquica

É a estrutura mais comum e permite rápida navegação, o usuário pode navegar por toda a hierarquia, tanto vertical como horizontalmente, a Figura 3.5 apresenta o esquema de como funciona a estrutura hierárquica.

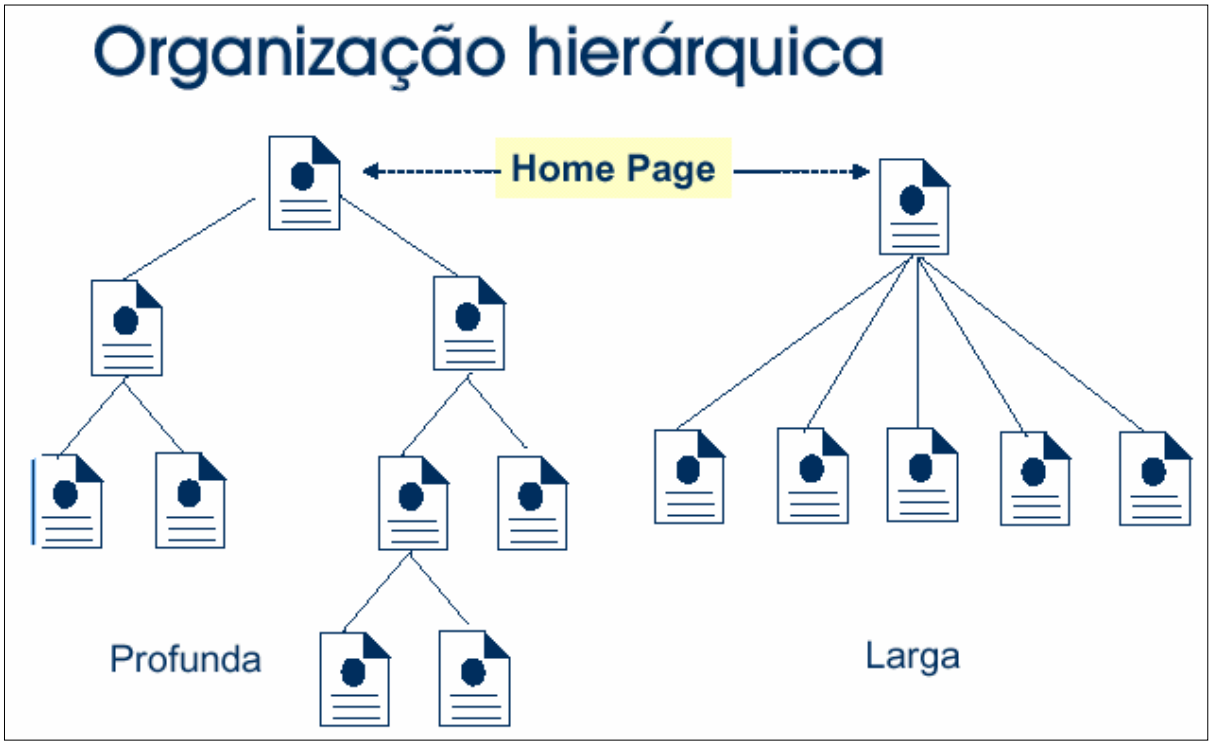

Figura 3.4: Esquema de organização hierárquica de *WebApps*.

3.1.3.1.3 Estrutura em rede ou “pura teia”

Similar ao modo como funciona a arquitetura de sistemas orientados a objetos, cada componente (páginas) é projetado de modo que possa passar comandos (via *links* de hipertexto) para qualquer outro componente do sistema, essa abordagem cria bastante flexibilidade de navegação, mas pode confundir o usuário, a Figura 3.3 apresenta o esquema de como funciona a estrutura em rede.

Figura 3.5: Esquema de organização em rede de *WebApps*.

Uma *WebApp* pode combinar diferentes estruturas. O objetivo deve ser criar a estrutura ideal para o conteúdo a ser apresentado.

3.1.3.2 Navegação

No projeto de navegação, são definidos os caminhos de navegação que permitem ao usuário ter aceso ao conteúdo e aos serviços da aplicação. Para isso, é preciso identificar as permissões de navegação para diferentes usuários e definir mecanismos para realizar a navegação.

Nessa etapa, é determinada a maneira como os *links* serão identificados. Dentre as opções, estão: textos, ícones e botões. O projetista deve escolher as opções apropriadas ao conteúdo, tendo como objetivo criar uma interface amigável.

De acordo com Pressman (2005), além de convenções sobre como utilizar a cor para ressaltar *links* e para indicar *links* já percorridos, um projeto de navegação deve incluir auxílios como mapas do *site*, sumários, índices e mecanismos de busca.

3.1.3.3 Interface

Em aplicações para *web*, a interface tem um papel muito importante, pois causa a primeira impressão no usuário. Independente do conteúdo, uma interface mal projetada pode desapontar o usuário, fazendo com que procure outro *site*. Conforme Pressman (2005, p.427),

“uma interface bem projetada melhora a percepção do usuário em relação ao conteúdo ou aos serviços oferecidos pelo *site*”.

O aspecto mais relevante à interface é aliá-la à usabilidade. Muitas vezes, o projetista se preocupa em utilizar as mais recentes inovações tecnológicas e oferecer muitas opções na aplicação, mas acaba criando uma aplicação difícil de usar.

Nielsen (2000) sugere as seguintes diretrizes para criar uma boa interface:

- Evitar erros do servidor;
- Criar páginas com pequenas quantidades de texto;
- Evitar avisos de “Em Construção”;
- Colocar informações importantes no topo, de forma que apareçam logo que a página for carregada;
- Disponibilizar menus e barras de navegação em todas as páginas em que o usuário irá navegar;
- Não contar com as funcionalidades do browser;
- Opções de navegação devem ser intuitivas.

3.1.4 Implementação

Essa atividade é também chamada de programação, construção ou codificação da *WebApp*. A implementação é o processo onde a aplicação é desenvolvida utilizando uma linguagem de programação e deve atender as especificações resultantes das atividades anteriores.

3.1.5 Testes

Assim como na engenharia de *software*, a engenharia para a *web* tem a atividade de teste, visando encontrar erros. Porém, os testes em *WebApps* são um desafio ainda maior, pois esses aplicativos podem ser acessados utilizando diferentes browsers, sistemas operacionais, plataformas de hardware e protocolos de comunicação.

Pressman (2005) apresenta uma abordagem que adota os princípios básicos para o teste de todo *software* e aplica estratégias e táticas que são recomendadas para sistemas orientados a objetos:

- Revisar o conteúdo a fim de descobrir erros de grafia, gramática, consistência do conteúdo, representações gráficas, dentre outros;
- Revisar o projeto de navegação para descobrir erros nos *links* e/ou na permissão de acesso de cada usuário;
- Testar cada página focando seu processamento;
- Testar a funcionalidade geral e conteúdo fornecido;
- Testar a compatibilidade da aplicação em diferentes configurações (diferentes *browsers*, sistemas operacionais, plataformas de *hardware* e protocolos de comunicação).

Por fim, é de grande valia submeter um grupo de usuários a testes de aplicação de forma monitorada, de modo a revisar todos os itens acima, bem como encontrar erros de desempenho e confiabilidade do aplicativo.

3.1.6 Instalação

A instalação visa à implantação do sistema para a utilização pelos usuários. Nessa atividade, também são realizadas as configurações, tanto de hardware como da aplicação (parâmetros iniciais), necessárias para que a aplicação tenha seu desempenho conforme o planejado.

Realiza-se, também, a verificação da infra-estrutura definida na etapa de análise e especificação de requisitos, envolvendo banco de dados, servidores, estrutura de rede para a distribuição da aplicação (internet, intranet ou extranet), entre outros.

3.1.7 Evolução e manutenção

De acordo com Leite (2002), a manutenção é a atividade destinada a assegurar a qualidade da aplicação durante a fase de operação.

A manutenção se faz necessária para que o sistema possa ser transportado para outras plataformas de *hardware* e de sistemas operacionais, para satisfazer novos requisitos dos

usuários ou para corrigir eventuais erros. Ela pode envolver a correção de erros não detectados durante os testes, revisão da funcionalidade, da interface com o usuário, da arquitetura da aplicação, entre outros, de forma que novos requisitos sejam satisfeitos.

3.2 Métodos de Desenvolvimento de Aplicações *Web*

Para gerenciar o desenvolvimento e manutenção de uma aplicação *web*, um método de desenvolvimento deve ser estabelecido, combinando técnicas e princípios tradicionais da engenharia de *software* com os aspectos específicos da *web* (BRAMBILLA et al., 2002), aspectos esses não abordados por métodos de desenvolvimento conhecidos, como o OMT (RUMBAUGH et al., 1991), ou linguagens de modelagem, como a UML (RUMBAUGH et al.,1998).

De modo geral, os métodos atuais de desenvolvimento *web* estendem métodos clássicos de desenvolvimento de *software* pela adição de algum modelo de navegação para especificar as características navegacionais de uma aplicação *web*. Um procedimento similar também é utilizado para a adaptação de linguagens de modelagem extensíveis, como a UML, ao contexto das aplicações *web*. Esses modelos definem visões navegacionais de um sistema e relacionam-se a grupos de usuários. Geralmente, descrições navegacionais são representadas por grafos que especificam visões sobre os dados de uma aplicação e funcionalidades definidas em modelos estruturais. Os nós desses grafos representam visões do sistema e podem ser relacionados por ligações navegacionais (*links*) (TORRES et al., 2004).

Nos últimos anos, surgiram diversos métodos que procuram adequar tecnologias conhecidas ao desenvolvimento de aplicações *web* (GRIFFITHS et al., 2002). Além disso, características como o curto prazo de tempo requisitado para o desenvolvimento de aplicações *web*, o imediatismo, podem sugerir a utilização de outros métodos, como os métodos ágeis (FOWLER, 2000), pois, entre seus princípios, encontra-se a entrega de versões do *software*, que funcionam adequadamente, em curtos períodos de tempo (CAGNIN et al., 2005).

Nas, Tabelas (3.1a, 3.1b, 3.1c) a seguir estão sintetizados os atributos dos principais métodos de desenvolvimento de aplicações *web*. Para melhor compreensão, na quarta coluna das tabelas estão numeradas as representações gráficas e na quinta coluna, a mesma numeração é utilizada para relacionar qual notação é utilizada para cada representação.

Tabela 3.1a: Comparação entre Métodos de Desenvolvimento: Atributos Gerais (DOMINGUES et al., 2007).

Método	Processo	Técnica	Representação Gráfica	Notação	Ferramenta
HDM	1. Esquema HDM 2. Instanciação do esquema HDM 3. Definição da semântica de navegação	E-R	1. Diagrama E-R 2. Diagrama E-R 3. Diagrama E-R	1. E-R 2. E-R 3. E-R	
RMM	1. Projeto E-R 2. Projeto de fatias 3. Projeto navegacional 4. Projeto de interface com o usuário 5. Projeto de protocolos de conversão 6. Projeto de comportamento em tempo real 7. Construção e teste	E-R	1. Diagrama E-R 2. Diagrama de fatias 3. Diagrama RMDM	1. E-R 2. própria 3. própria	
OOHDM	1. Projeto conceitual 2. Projeto navegacional 3. Projeto de interface abstrata 4. Implementação	OO	1. Diagrama de classes 2. Classes navegacionais e esquema navegacional 3. ADVs e ADV-Charts	1. UML 2. própria 3. própria	OOHDM-Web
HMBS	1. Modelagem conceitual 2. Modelagem navegacional 3. Modelagem de interface 4. Implementação e teste	OO e estados	1. Modelo de objetos e modelo de fatias 2. Contextos de navegação modelo navegacional de tipos e de instâncias 3. Canais de apresentação	1. *Fusion* 2. *StateCharts* 3. própria	*HyCharts* e *WebScharts* (HMBS/M estendido)

Tabela 3.1b: Comparação entre Métodos de Desenvolvimento: Atributos Gerais (DOMINGUES et al., 2007).

Método	Processo	Técnica	Representação Gráfica	Notação	Ferramenta
W2000	1. Análise de requisitos 2. Projeto de evolução de estados 3. Projeto de hipermídia 4. Projeto funcional 5. Projeto de visibilidade	E-R e OO	1. Casos de uso 2. Casos de uso estendido 3. Esquemas e diagramas de classe 4. Cenários e diagramas de interação 5. Visões do sistema	1. UML 2. UML 3. HDM; e UML 4. UML 5. própria	
UWE	1. Modelagem estrutural 2. Modelagem de hipertexto 3. Modelagem de apresentação 4. Modelagem de personalização	E-R	1. Organização conceitual dos dados; e diagrama de classes 2. Modelo de composição; e de navegação 3. Modelo navegacional de espaço; e de estrutura 4. Modelo de entidade pré-definida	1. E-R 2. própria 3. própria 4. própria	WebRatio
WAE	1. Gerenciamento de projeto 2. Obtenção de requisitos 3. Análise 4. Projeto 5. Implementação 6. Teste 7. Implementação	OO	1. Plano de projeto; de iteração; e de gerenciamento de alteração 2. Documento de declaração de visão 3. Modelo conceitual, diagramas de seqüência, de estado 4. Diagramas de classe, de seqüência	1. própria 2. própria e UML 3. UML 4. UML	
OO-H	1. Modelagem Conceitual 2. Projeto de Navegação 3. Projeto de Interface	OO	1. Diagrama de classes 2. Diagrama de Acesso navegacional 3. Diagrama abstrato de apresentação	1. UML 2. própria 3. própria	Oliva NOVA

Tabela 3.1c: Comparação entre Métodos de Desenvolvimento: Atributos Gerais (DOMINGUES et al., 2007).

Método	Processo	Técnica	Representação Gráfica	Notação	Ferramenta
OOWS	1. Modelagem Conceitual 2. Desenvolvimento da Solução	OO	1. Diagrama de Caso de Uso Cenários 2. Diagrama de Classes 3. Diagrama de Estado Diagrama de Seqüência 4. Diagrama Navegacional *(Navigational Map*)	1. UML 2. UML 3. UML 4. UML	Oliva NOVA
SWM	1. Planejamento 2. Análise 3. Projeto 4. Implementação 5. Manutenção	DFD	1. Especificação de projeto conceitual 2. Especificação de projeto do site	1. DFD 2. DFD	ASCENT
ECO	1. Aquisição 2. Entrega 3. Encerramento	OO	1. Mapa de Navegação 2. Modelos de Grade 3. Diagrama de Componentes 4. Modelo de Domínio	1. própria 2. UML 3. UML 4. UML /*Domain Neutral Component*	

Observa-se que, de um modo geral, os métodos abrangem fases similares, como análise inicial de seu domínio, seguido de projeto na estrutura e navegação da aplicação e, por fim, elaboração de interface da aplicação e sua construção e teste.

Para melhor conhecimento e prática do método selecionado, foi realizado um estudo de caso. Esse estudo consistiu na modelagem de uma mesma aplicação utilizando o método, o que permitiu observar as etapas de produção de uma *WebApp*.

4 ESTUDO DE CASO

O método escolhido para o estudo prático foi o WAE pelo fato de atender plenamente a maioria dos atributos que são determinantes para um bom projeto de desenvolvimento de uma aplicação *web*. Esses atributos, citados na seqüência, subdividem-se em dois grandes grupos, a saber: características de notação e características de modelo de engenharia.

As de notação são características que um método deve possuir para ser eficiente, prático e facilmente aprendido. Um método deve ser eficiente, pois deve ser capaz de representar um problema do mundo real, nos aspectos relevantes que esse problema venha a apresentar. Um método deve ser prático, a fim de que os projetistas e outros envolvidos em sua produção sejam capazes de construir modelos tanto com o auxílio de ferramentas CASE como construídos à mão. Quanto à facilidade de aprendizagem, por mais relativa que seja, essa característica ajuda os envolvidos diretos ou indiretamente no processo de desenvolvimento.

As características de modelos de engenharia representam características que um método deve possuir para que forneça recursos para a produção de uma aplicação com qualidade como, por exemplo, capacidade de abstração, previsibilidade e baixo custo, entre outras.

4.1 Características do Modelo de Desenvolvimento

4.1.1 Características de notação

4.1.1.1 Desenho à mão livre

O método não depende exclusivamente de uma ferramenta CASE para modelagem. Possibilita a criação de rascunhos rápidos.

4.1.1.2 Simplicidade de símbolos

Juntamente com o critério anterior, essa característica facilita o desenho à mão livre, a visualização e a compreensão de modelos por envolvidos não técnicos.

4.1.1.3 Diferenciação não sutil de símbolos

O reuso de símbolos com modificações sutis pode confundir tanto o projetista como usuários que não são desenvolvedores.

4.1.1.4 Imagens monocromáticas

Imagens monocromáticas trabalham com formas e símbolos que permitem o envio de faxes e cópias, sem comprometer o modelo desenhado e é mais facilmente reproduzido.

4.1.1.5 Não sobrecarga de símbolos

Símbolos usados em mais de um modelo provocam sobrecarga nos mesmos, dificultando o trabalho tanto de construção como de compreensão do modelo.

4.1.1.6 Descrição visual

Característica imprescindível no método, pois representa diagramas e gráficos que podem ser construídos nos modelos.

4.1.1.7 Descrição textual

Auxilia a compreensão de modelos, com o relato de detalhes e observações. Geralmente é acompanhada de uma descrição visual.

4.1.1.8 Modelagem conceitual

Compreensão dos conceitos e elementos que compõem o problema e suas relações. Compreensão do domínio da aplicação.

4.1.1.9 Modelagem navegacional separada da interface

A modelagem navegacional deve ser separada da modelagem da interface. Na modelagem navegacional, é modelado como a informação navega entre os componentes da aplicação, sem a preocupação de como tal informação é apresentada e posicionada ao usuário final, tarefa essa realizada pela modelagem da interface. Salienta-se que modelando separadamente essas duas características de uma aplicação, projetistas ou equipes diferentes podem trabalhar em paralelo, uma com a modelagem da navegação e outra com a modelagem de interface.

4.1.1.10 Consideração de múltiplos papéis de usuários

Característica pertinente às aplicações *web* que são utilizadas por usuários heterogêneos, com diferentes habilidades e propósitos de uso.

4.1.1.11 Propõe processo

Além de fornecer a técnica para modelagem, o método propõe um processo associado que possui fases claras e definidas, não só para modelagem e projeto da aplicação, como também para outras fases conhecidas da engenharia de *software*, como análise de requisitos, desenvolvimento, testes, entrega.

4.1.1.12 *Links* semânticos

Links fazem parte da estrutura básica de uma aplicação *web* e devem ser utilizados corretamente, obedecendo ao contexto da informação que está sendo apresentada para o usuário.

4.1.1.13 Desenho de telas

Recurso que realça a compreensão de elementos da interface.

4.1.1.14 Modelagem de interface

Modelagem dos componentes que apresentam a informação, bem como seu posicionamento na tela para o usuário final. Considerando as aplicações *web*, a modelagem de interface deve ser capaz de tratar e representar conteúdo hipermídia e multimídia.

4.1.1.15 Modelagem independente de tecnologia

É desejável que um método seja independente de qualquer tecnologia adotada na codificação da aplicação.

4.1.2 Características de Modelos de Engenharia

4.1.2.1 Abstração

Capacidade de remover ou esconder detalhes irrelevantes para um dado ponto de vista, constitui um recurso para trabalhar com complexidades.

4.1.2.2 Compreensibilidade

Apresentação de modelos que diretamente remetem à compreensão e assim provê a redução do esforço intelectual.

4.1.2.3 Conformidade

Representação da aplicação em conformidade com o modelo real (de acordo com o resultado esperado no espaço das soluções possíveis).

4.1.2.4 Previsibilidade

Possibilidade de prever características da aplicação, que possam ser alteradas tanto no decorrer da fase de projeto, como durante o ciclo de vida da aplicação.

4.1.2.5 Custo-baixo

A criação de modelos de baixo custo facilita a recriação e correção de etapas já realizadas, relacionado com a característica 4.1.2.1.

4.1.2.6 Método iterativo

Aceita a revisitação a um determinado modelo ou uma determinada fase da modelagem, com o intuito de corrigir ou refinar tal modelo.

4.1.2.7 Abordagem *top-down*

O método permite a utilização da estratégia *top-down*. A partir de conceitos mais abstratos, é possível refiná-los em uma granularidade maior, o que permite a representação de detalhes.

4.1.2.8 Integração de aspectos funcionais com aspectos de informação

O modo como as informações apresentadas aos usuários refletem as funções que as produzem é um ponto importante a ser verificado. Uma arquitetura comumente utilizada, a MVC (Seção 2.6) conecta esses dois aspectos de modo simples e modular, permitindo o trabalho colaborativo.

4.1.2.9 Uso de padrões de projeto

O uso de padrões já utilizados e comprovados pela indústria e academia fornece subsídios para desenvolvimento ágil e prototipação rápida de aplicações.

4.2 O Modelo WAE

O *Web Application Extension* (WAE) (CONALLEN, 2002) é um conjunto de novos elementos que estendem a linguagem de modelagem UML.

Segundo Conallen (2002), a construção de uma aplicação *web* inicia-se com uma análise do problema, desenvolvimento de um modelo do domínio (por meio da notação UML), desenvolvimento de um documento de visão (escopo e proposta do projeto) e desenvolvimento do plano de projeto. No plano de projeto deve existir um plano de iteração, que descreve detalhadamente as atividades que são esperadas no decorrer do projeto, os artefatos que devem ser produzidos ao final de cada atividade e seus critérios de aprovação. Após o desenvolvimento desses produtos de trabalho, seguem as etapas para o desenvolvimento da arquitetura e projeto da aplicação, seguido de sua implementação e teste.

Para o desenvolvimento da arquitetura são utilizados os recursos disponíveis na UML, como o diagrama de casos de uso, de classes, de atividade, de seqüência. Além desses, há um outro recurso: o modelo UX (*User eXperience*). No modelo UX é possível modelar o fluxo de informações da aplicação, por meio da determinação das principais rotas de navegação entre as páginas *web* que compõem a aplicação, além de gerenciar e organizar a estrutura do conteúdo das páginas.

O modelo UX é composto por definições de telas, roteiros e mapas de navegação. A principal entidade a ser modelada é a tela. Uma tela é um elemento que é apresentado ao usuário e contém a infra-estrutura padrão da interface com o usuário, como menus e controles, assim como o conteúdo referente ao negócio propriamente modelado. Salienta-se que o conceito de tela não deve ser confundido com o conceito de páginas *web*, que são os elementos que produzem as telas. A representação desse elemento no modelo UX é uma classe dada pelo estereótipo <<screen>>. Além da representação escrita há também a representação gráfica por meio de um ícone como mostrado na Figura 4.1.

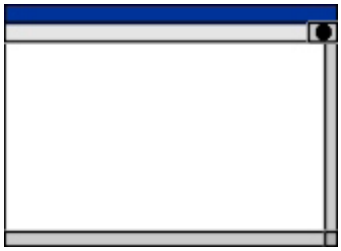

Figura 4.1: Representação em ícone de estereótipo criado no método WAE.

Entre os elementos tela, há o fluxo de navegação que é a transição de uma tela à outra. Caminhos entre telas compõem cenários que formam elementos arquitetônicos importantes, em conjunto à análise de requisitos. No modelo UX, tais caminhos são modelados por associações entre as classes estereotipadas e podem possuir nomes que ajudam a explicar a intenção do usuário ou a ação que resultou na navegação de uma tela para outra. Além do estereótipo <<screen>>, existem o estereótipo <<inputform>>, utilizado para representar a tela de entrada de dados do usuário, e o estereótipo <<screencompartment>>, utilizado para representar porções reutilizáveis de tela ou sub-telas. Os roteiros do modelo UX mapeiam um cenário de um caso de uso utilizando-se das telas previamente definidas em conjunto com as associações e o mapa de navegação apresenta-se como um diagrama similar ao roteiro, porém simplificado, pois não inclui detalhes, por exemplo, como as propriedades das telas. O WAE ainda inclui estereótipos para o chamado modelo de projeto. Esse modelo consiste em projetar componentes que vão desde código executado pelo servidor até elementos internos às páginas.

Além dessas atividades, o método é composto por atividades de apoio como, por exemplo, a gerência de configuração, que gerencia as versões dos artefatos produzidos. Permite também um desenvolvimento iterativo ao propor revisões e melhorias no plano de iteração inicial, durante o ciclo de desenvolvimento.

O método WAE possui uma fase de análise longa, pois utiliza além dos casos de uso, diagramas de atividades que complementam a análise, procurando compreender o fluxo pelo qual os possíveis uso da aplicação são realizados.

Apesar de o modelo UX possuir um diagrama estendido dos diagramas de classes convencionais, tal diagrama é desconexo com qualquer diagrama de classes que represente o domínio do sistema, ou seja, não há representação ou instância de alguma classe do diagrama de classes. Isso acontece, pois o diagrama do modelo UX representa exclusivamente as páginas pelas quais o usuário pode navegar e elementos inerentes a ela, como divisão de

blocos da tela, formulário, campos de entrada e outros elementos presentes em uma página *web*. Por esse modelo, é claro perceber a visão do autor, Conallen, em oferecer um modelo destinado a uma equipe especializada na construção de página *web* composta por *web designers* e profissionais com experiência em hipermídia. Para aplicações onde o projetista ou uma equipe de projetistas trabalham em todas as fases da elaboração de uma aplicação, é necessário que para a construção do modelo UX esses profissionais possuam já uma visão da aplicação em termos das páginas que a compõem, a fim de que construa corretamente o modelo. Um ponto controverso a essa observação, é que o mesmo profissional que construa o modelo UX e detenha conhecimento em hipermídia e áreas afins, conheça a própria estrutura de um diagrama de classes, pois uma vez que o diagrama existente nesse modelo estende um diagrama de classes, esse diagrama estendido contém toda a semântica e recursos que existem no original. Complementando as descrições de tela existentes no diagrama estendido de classes, é necessário para o modelo UX que sejam construídos os roteiros guiados que auxiliam a compreensão das definições das telas e o fluxo de informação da aplicação.

4.3 Etapas Genéricas de Desenvolvimento

4.3.1 Definição do problema

A aplicação desenvolvida no contexto do presente trabalho foi o *site* flowJava. O intuito da criação do *site* é explicitar as etapas produtivas envolvidas no processo de produção de uma aplicação *web*, bem como oferecer à comunidade acadêmica e a interessados, informações sobre o tema: *workflow* e Java. Vale ressaltar que o *site* não será referência no assunto java isolado, tendo em vista a existência de milhares de aplicações *web* com esse foco.

Os usuários em potencial do *site* são: pessoas com uma certa familiaridade no uso de computadores, programadores, analistas de negócio, pessoas que trabalham com BPM (*Business Process Management*).

4.3.2 Análise e especificação de requisitos

O flowJava é uma aplicação *web*, que por meio de um único ponto de acesso integra usuários e serviços tais como, compartilhamento e utilização de informação:

- artigos - oferece à comunidade acadêmica, conhecimento obtido sobre *workflow* e java, páginas *web*;

- tutoriais – textos e vídeos;
- fórum para suporte a dúvidas relacionadas a *workflow* e java;
- ferramentas – *plugins* que rodam em IDEs (*Integrated Development Environment*) java como Eclipse, por exemplo.

O *web site* permite também a interação de usuários, de modo que esses possam incluir informações a respeito de ferramentas, técnicas e informações em geral de *workflow* e java, além de realizar consultas sobre o conteúdo existente no *site*, inclusão de comentários sobre os tópicos incluídos, a fim de abrir diálogo e comunicação entre usuários também é possível de se realizar.

Concomitante à prestação dos serviços mencionados no parágrafo anterior é necessário que a *WebApp* forneça suas funcionalidades de forma segura, de modo que facilite a visualização de informação relevante a diferentes perfis de usuários. O *site* trabalha com recursos técnicos como: manipulação de arquivos, integração com ferramentas externas, lógica de negócio apurada e uso de multimídia para interfaces ricas.

Tópicos do fórum, artigos e ferramentas, por exemplo, só serão exibidos aos visitantes e usuários, mediante aprovação do administrador - é interessante mencionar nesse ponto, as diferenças entre esses dois perfis de clientes (visitantes e usuários) do *site*, quais sejam: visitantes, o nível de acesso resume-se somente a visualização; usuário faz *login* na área determinada do *site* possibilitando-o a cadastrar artigo, iniciar um tópico para discussão, realizar o *upload* de alguma ferramenta, dentre outras funcionalidades.

4.3.2.1 Especificação de requisitos

O ambiente do *site* é constituído por três módulos:

- Módulo de conhecimento;
- Módulo de colaboração;
- Módulo de consulta.

O módulo de conhecimento refere-se a comentários que usuários cadastram sobre a qualidade de tutoriais, ferramentas entre outros tópicos do fórum.

No módulo de colaboração, usuários inserem dados como: artigos, tutoriais, informações sobre o uso de ferramentas, etc.

Finalmente, no módulo de consulta são executadas buscas sobre o conhecimento armazenado no *site*, de acordo com chaves de pesquisa.

Além da descrição das funcionalidades e módulos do *web site*, há também um conjunto de papéis para usuários, que possuem diferentes níveis de permissão e acesso. Esses papéis são:

- Visitante: usuário que acessa o *site* para obter informações sobre o conteúdo existente e, eventualmente, pode solicitar permissão para acessar módulos internos ao *site*;
- Usuário cadastrado: usuário que já possui permissão para acesso aos módulos de conhecimento, colaboração e de consulta, pode ver as informações existentes e comunicar-se com os demais usuários do *site*;
- Administrador: usuário *master* que possui permissão em todos os módulos citados, além de ser o responsável por gerenciar todo o conteúdo do *site*.

4.3.2.2 Requisitos técnicos

A arquitetura da aplicação é semelhante à apresentada na Seção 2.6, ou seja, um cliente pelo navegador, acessa a página do flowJava no servidor flojava.com e aguarda uma resposta. O servidor então responde os pedidos do *browser*. Verifica se o endereço existe, encontra os arquivos necessários, executa as instruções apropriadas e retorna os resultados. O *browser* recebe os dados do servidor na linguagem HTML, interpreta essas instruções e exibe os resultados para o usuário.

4.3.2.2.1 Para a hospedagem do site

A aplicação roda em um servidor na plataforma Windows, utilizando o banco de dados MySQL e a linguagem de programação PHP 5.

4.3.2.2.2 Para o desenvolvimento do *site*

Foi utilizado o CMS (*Content Management System*) Joomla, na versão 1.5.15, que é um sistema gerenciador de conteúdo, o qual é escrito em PHP e roda em servidores Apache e ISS e banco de dados MySQL.

Principais características:

- Código aberto (*OpenSource*);
- Sistema simples de fluxo de aprovação;
- Gerenciamento de banners;
- Sistema para a publicação de conteúdo;
- Sumário de conteúdo em RSS;
- Sistema de busca avançado;
- Hierarquia para grupos de usuários;
- Estatísticas de visitantes;
- Sistema de enquetes;
- Sistemas de índices de avaliação;

Além do Joomla, foi utilizado também o Filezilla, na versão 3.3.0, que é um cliente FTP para a transferência das páginas desenvolvidas localmente para o *host* de hospedagem, também foi utilizado WAMP5, na versão 2.0, que é um pacote de programas que instala automaticamente o Apache 1.331, PHP5, MySQL *database*, PHPmyadmin e SQLitemanager.

4.3.2.3 Requisitos funcionais

4.3.2.3.1 Modulo de conhecimento

No módulo de conhecimento, o usuário insere os comentários sobre os tutoriais, artigos e ferramentas, avaliando a qualidade do mesmo. Os requisitos definidos para esse módulo são os seguintes:

1. O sistema deve permitir o cadastro de comentários de tutoriais e ferramentas, por meio da inclusão das informações no formulário de cadastro de comentários.

    1.1. Nome do artigo, tutorial ou ferramenta a ser comentado;

    1.2. Texto do comentário;

    1.3. Data – inclusão automática;

1.4. Usuário – inclusão automática.

4.3.2.3.2 Módulo de colaboração

1. O sistema deve permitir o cadastro de usuários a fim de que os mesmos quando logados, possam fazer *downloads*, postar informações, etc; por meio da inclusão das informações do formulário de cadastro de usuários, tais como:

1.1. Nome;

1.2. Senha;

1.3. Endereço (Cidade, Estado);

1.4. Se deseja receber informações sobre itens cadastrados.

2 O sistema deve permitir o cadastro de ferramentas para *download*, por meio da inclusão das informações no formulário de cadastro de ferramentas.

2.1. Nome da ferramenta;

2.2. Autor do *post*;

2.3. Descrição resumida;

2.4. Categoria – *WorkFlow* ou Java.

3. O sistema deve permitir o cadastro de artigos sobre *WorkFlow* e Java, por meio da inclusão das informações do formulário de cadastro de artigos:

3.1. Título do artigo;

3.2. Categoria;

3.3. Texto com o artigo.

4. O sistema deve permitir o cadastro de materiais de apoio relacionados ao estudo/utilização de ferramentas, critérios de uso de aplicações de *WorkFlow* e Java, com a inclusão das seguintes informações:

4.1 Título;

4.2 Autor;

4.3 Data;

4.4 Descrição;

4.5 Palavras Chave;

4.6. Classificação, dentre as opções: manuais, artigos e tutoriais;

4.7. *Link* deve permitir o direcionamento ao site que disponibiliza o material;

4.8. *Upload* para o caso de ferramentas com no máximo 2MB que ficarão hospedadas no servidor do *site*.

5. O sistema deve permitir a submissão do material de apoio segundo um processo de aprovação. O processo de aprovação consiste dos seguintes passos:

5.1. O usuário cadastrado submete um conteúdo no *site*;

5.2. O sistema categoriza o novo conteúdo com o *status* de "aguardando aprovação" e não permite a visualização por nenhum usuário;

5.3. O sistema notifica o administrador que um novo conteúdo foi adicionado;

5.4. O administrador altera o *status* do conteúdo para "aprovado" e o sistema permite a visualização para todos os usuários.

6. O sistema deve manter um histórico (*log*) que registre as operações realizadas pelos usuários.

7. O sistema deve possuir em cada formulário de cadastro um campo, "categoria", que permita ao usuário relacionar o item cadastrado aos demais itens já incluídos na base de dados.

8. O sistema deve notificar aos usuários que tenham escolhido receber informação a respeito da inclusão de novos conteúdos.

9. O sistema deve permitir que os usuários participem da área do fórum existente no portal.

10. O sistema deve permitir que o usuário escolha os fóruns que quer participar, inserindo comentários, adicionando respostas e criando novos tópicos para discussão.

11. O sistema deve permitir que o usuário envie *e-mails* aos demais usuários do sistema.

12. O sistema deve permitir que os usuários registrem opiniões a respeito do conteúdo disponível e os tornem visíveis aos demais usuários.

4.3.2.3.3 Módulo de consulta

Para o módulo de consulta, os requisitos abaixo são exigidos:

1. O sistema deve possuir um formulário para a busca, que permita a busca por:

    1.1. Palavra chave;

    1.2. Título;

    1.3. Autor;

    1.4. Tipo de Material.

2. O sistema deve ordenar e apresentar os resultados de busca de acordo com os itens de cada conteúdo.

3. O sistema deve permitir que o usuário visualize o conteúdo desejado e permita a realização de *download* desse conteúdo, de acordo com os critérios de segurança e acesso.

4. O sistema deve permitir que, na visualização de um conteúdo, existam *links* para os conteúdos relacionados.

Os requisitos para as tarefas administrativas são:

1. O sistema deve permitir o cadastro de grupos de usuário no sistema. Cada grupo possui um nível de acesso diferente para cada parte do portal. Os grupos são: usuário cadastrado e administrador.

O visitante não está relacionado a nenhum grupo, já que não possui vínculo cadastral com o sistema;

2. O sistema deve permitir o acesso de usuários não cadastrados, que podem visualizar conteúdos não restritos (definidos pelo administrador) do sistema.

4.3.3 Desenvolvimento do *web site*

Os diagramas desenvolvidos utilizando UML são apresentados a seguir:

- Diagrama de Caso de Uso: mostra atores (pessoas ou outros usuários do sistema), casos de uso (os cenários onde eles usam o sistema), e seus relacionamentos;
- Diagrama de Classe: mostra classes e os relacionamentos entre elas;
- Diagrama de Seqüência: mostra objetos e uma seqüência das chamadas do método feitas para outros objetos.

4.3.4 Estrutura global da aplicação

A estrutura navegacional do flowJava é a hierárquica, por permitir o acesso rápido a outras páginas, podendo navegar tanto vertical quanto horizontalmente.

4.3.5 Navegação

Os *links* do *site* são identificados por textos sublinhados. No topo do *site* é exibido o caminho percorrido pelo usuário até chegar à página de navegação, facilitando assim, a localização do usuário no *site*.

4.3.6 Interface

Projetada em tons de preto e branco para facilitar a leitura e minimizar o esforço ocular.

As mensagens de erro são sinalizadas em vermelho.

4.3.7 Implementação

A implementação foi realizada localmente e enviada para o servidor por um gerenciador de FTP (FileZilla).

4.3.8 Testes

Foram simulados vários cliques nas páginas, a fim conferir a corretude dos *links* e concomitante a isso, analisou-se o desempenho quanto ao tempo de carregamento das páginas.

O *web site* foi testado em navegadores diferentes, a fim de verificar como as páginas se apresentam aos visitantes.

4.3.9 Evolução e manutenção

Por ser versão 1.0 do *web site* não foi realizada nenhuma evolução quanto à *template*, *banners*, etc. Contudo, no aspecto de manutenção, é realizada semanalmente a aprovação de conteúdos submetidos por usuários disponibilizando-os para a comunidade.

4.4 Diagramas do Estudo de Caso

4.4.1 Diagrama de caso de uso

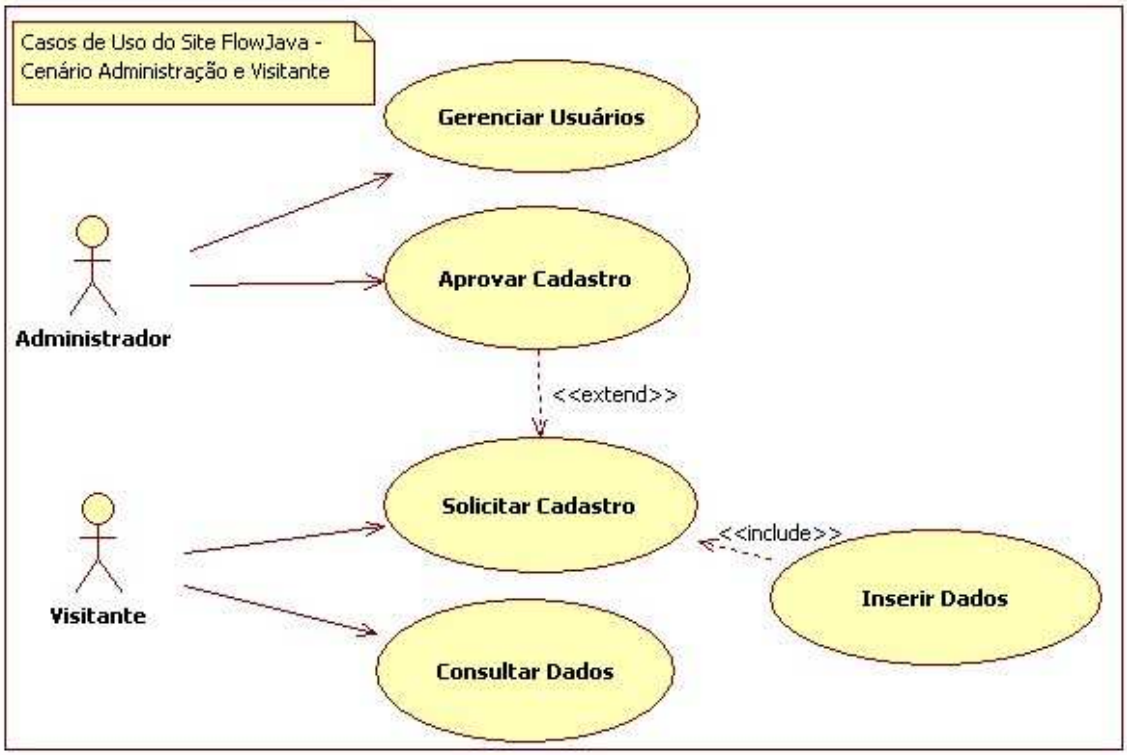

Figura 4.2: Caso de uso funcional - Administrador e visitante.

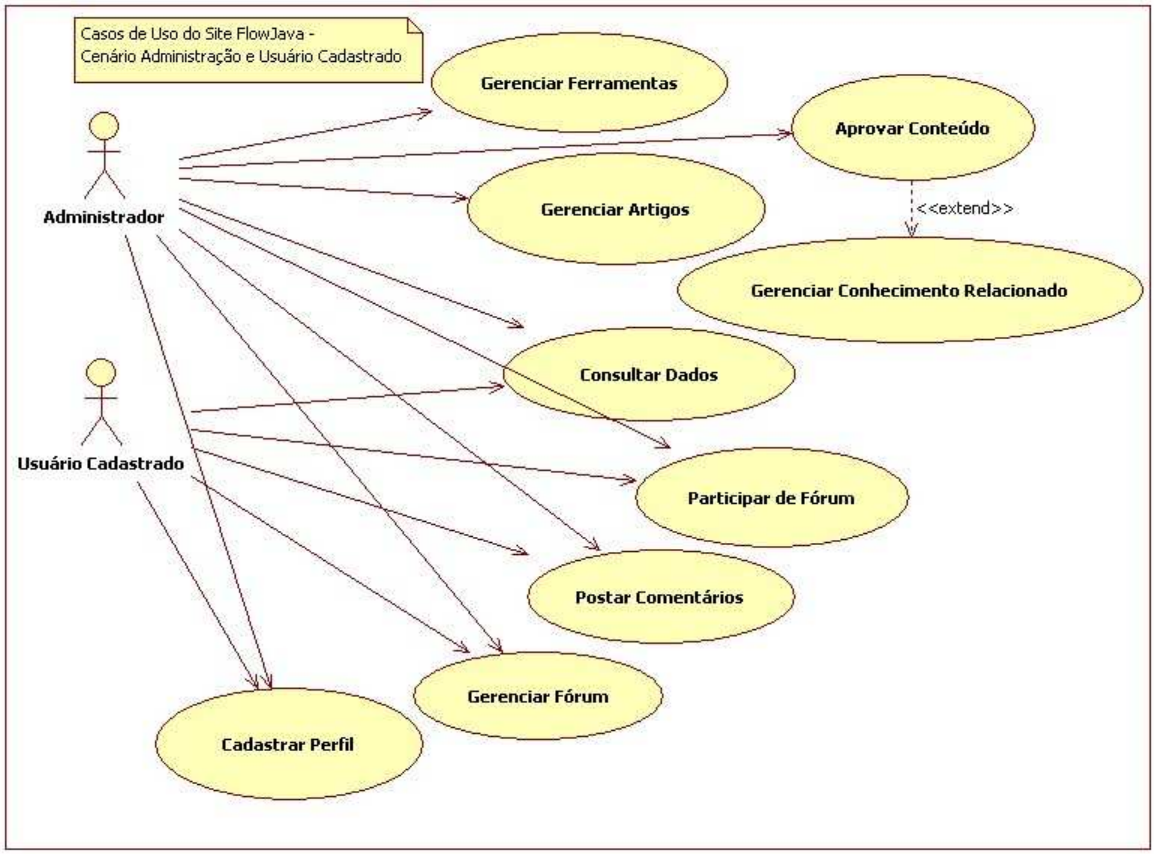

Figura 4.3: Caso de uso funcional - Administrador e usuário cadastrado.

4.4.2 Diagrama de atividades

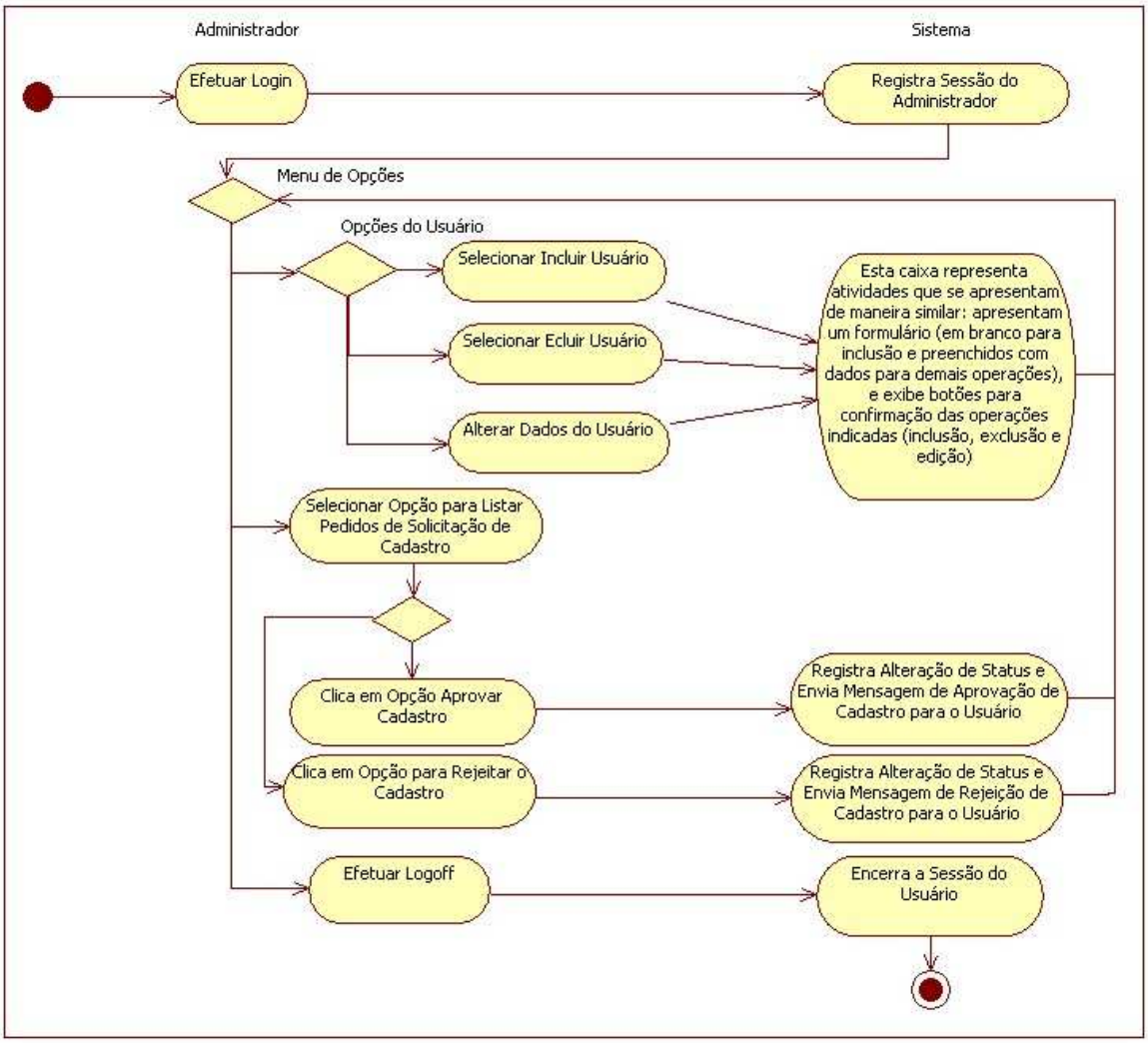

Figura 4.4: Diagrama de atividades do administrador do *site* I.

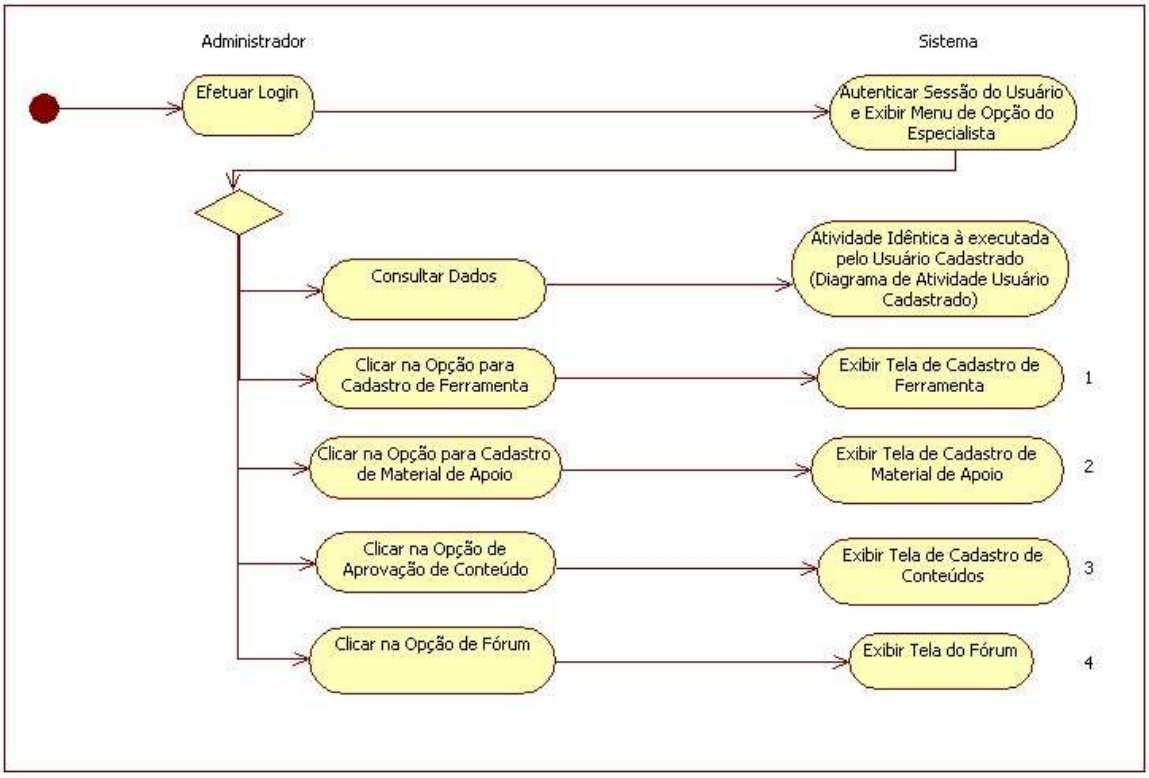

Figura 4.5: Diagrama de atividades do administrador do *site* II.

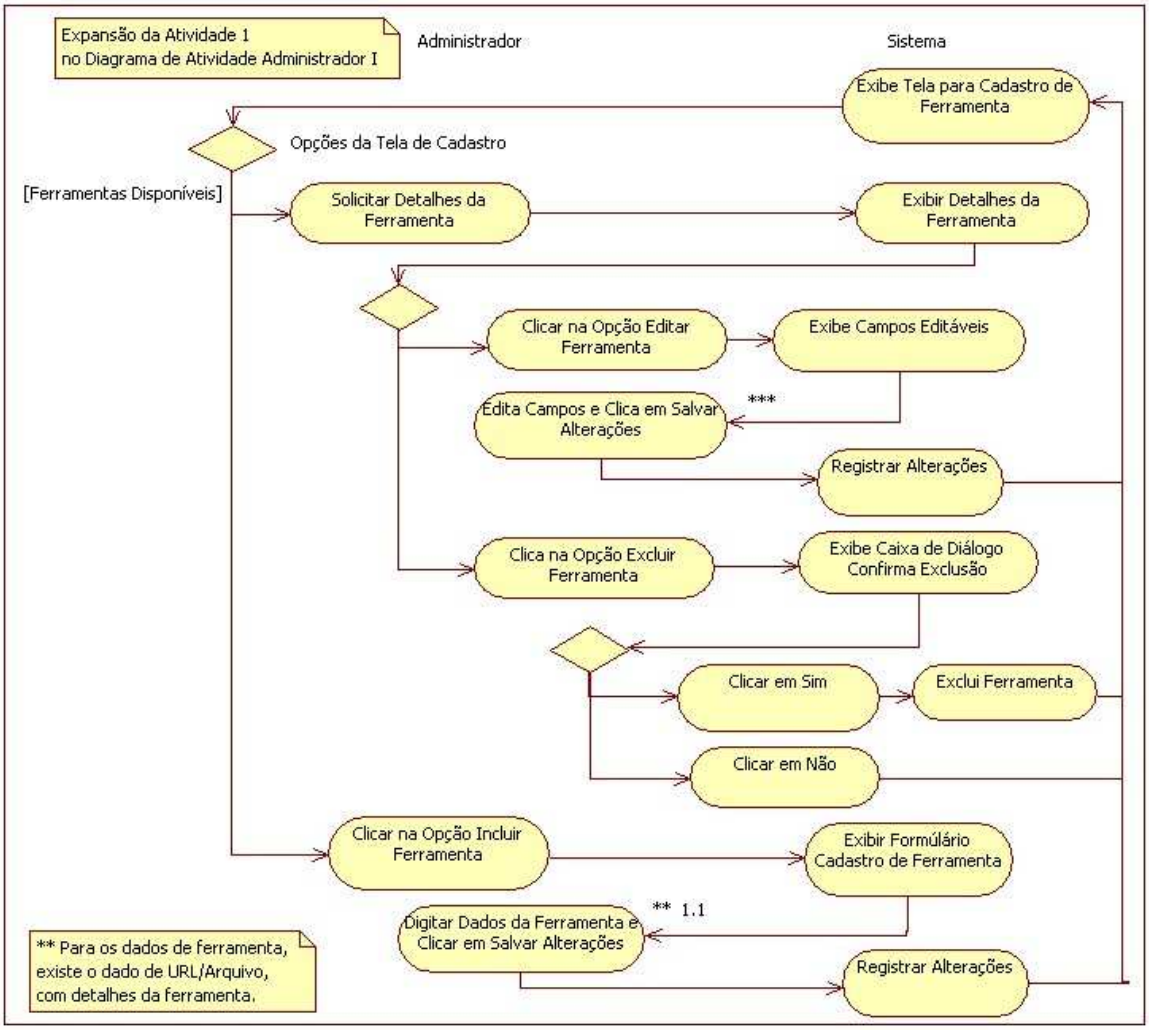

Figura 4.6: Diagrama de atividades do administrador do *site* III.

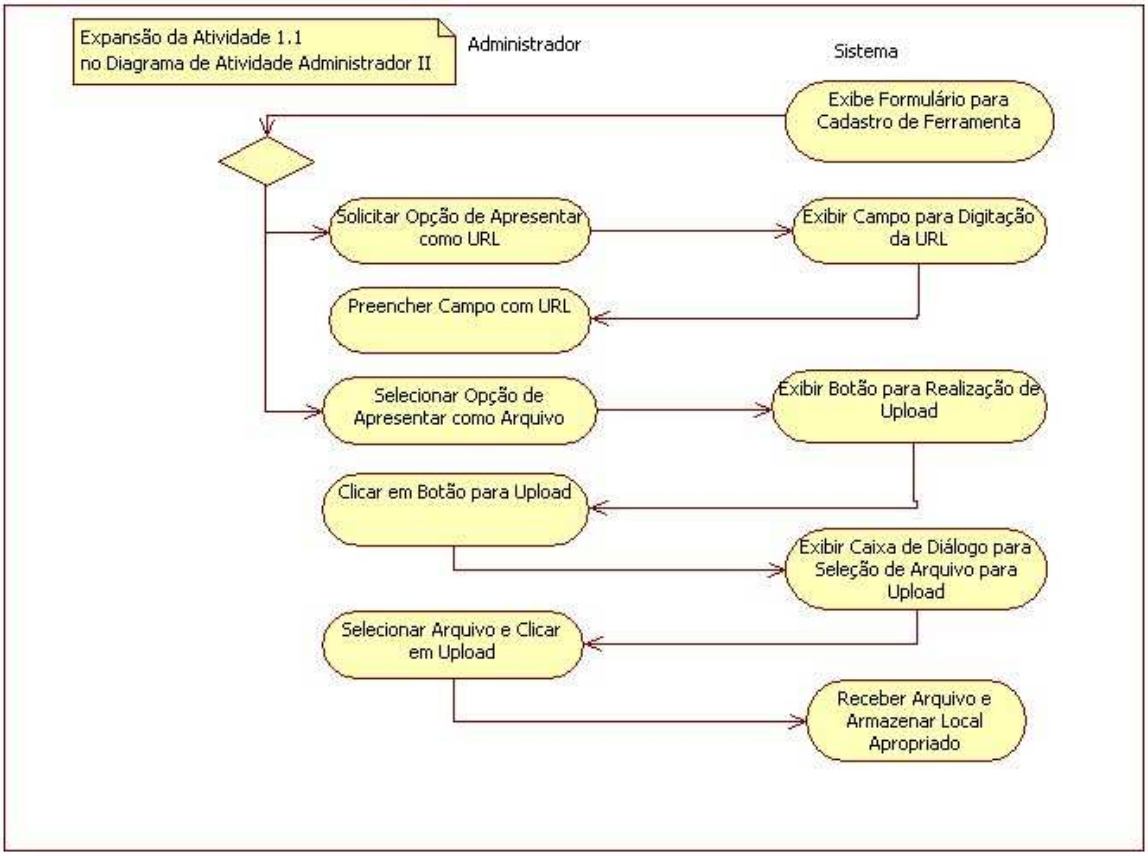

Figura 4.7: Diagrama de atividades do administrador do *site* IV.

4.4.3 Diagrama de transição de estados

Figura 4.8: Diagrama de evolução de estados para requisição de cadastro.

Figura 4.9: Diagrama de evolução de estados para submissão de conteúdo.

4.4.4 Diagrama de classes

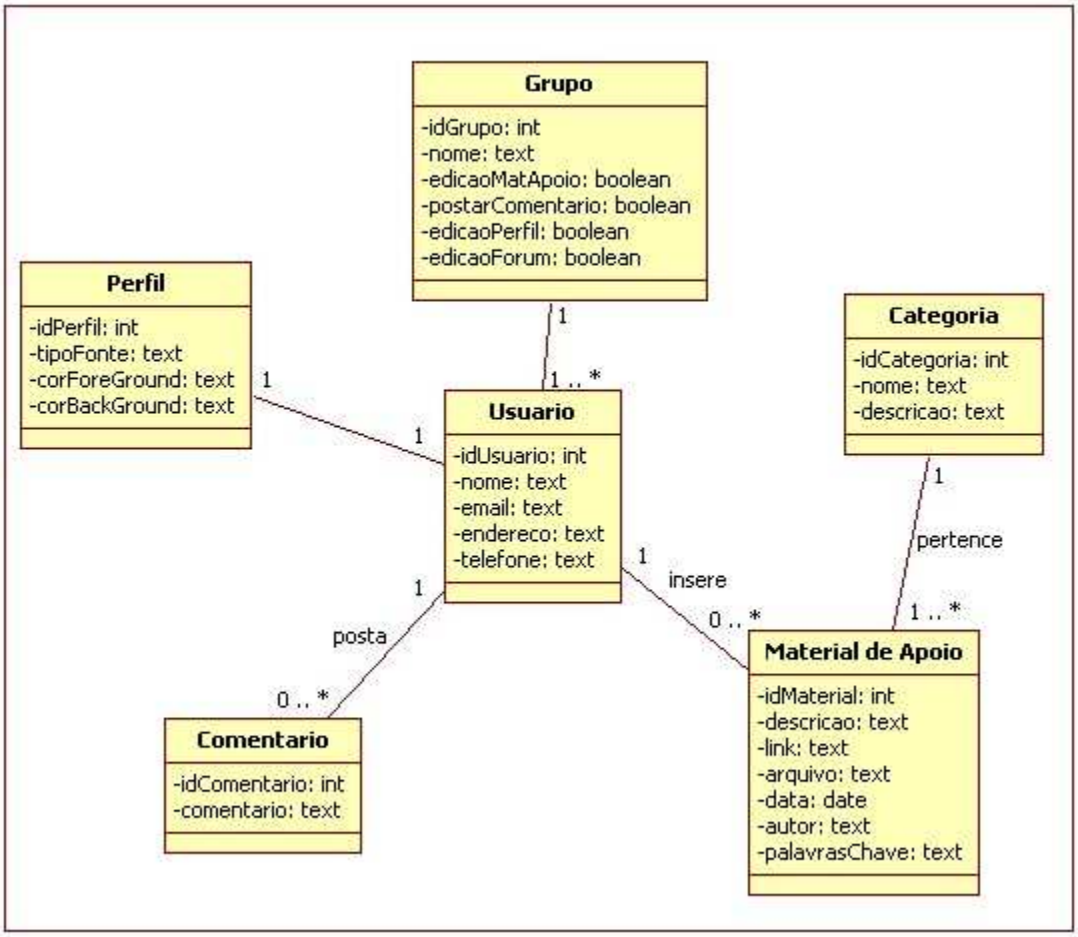

Figura 4.10: Diagrama de classes do *site*.

4.4.5 Diagrama de navegação

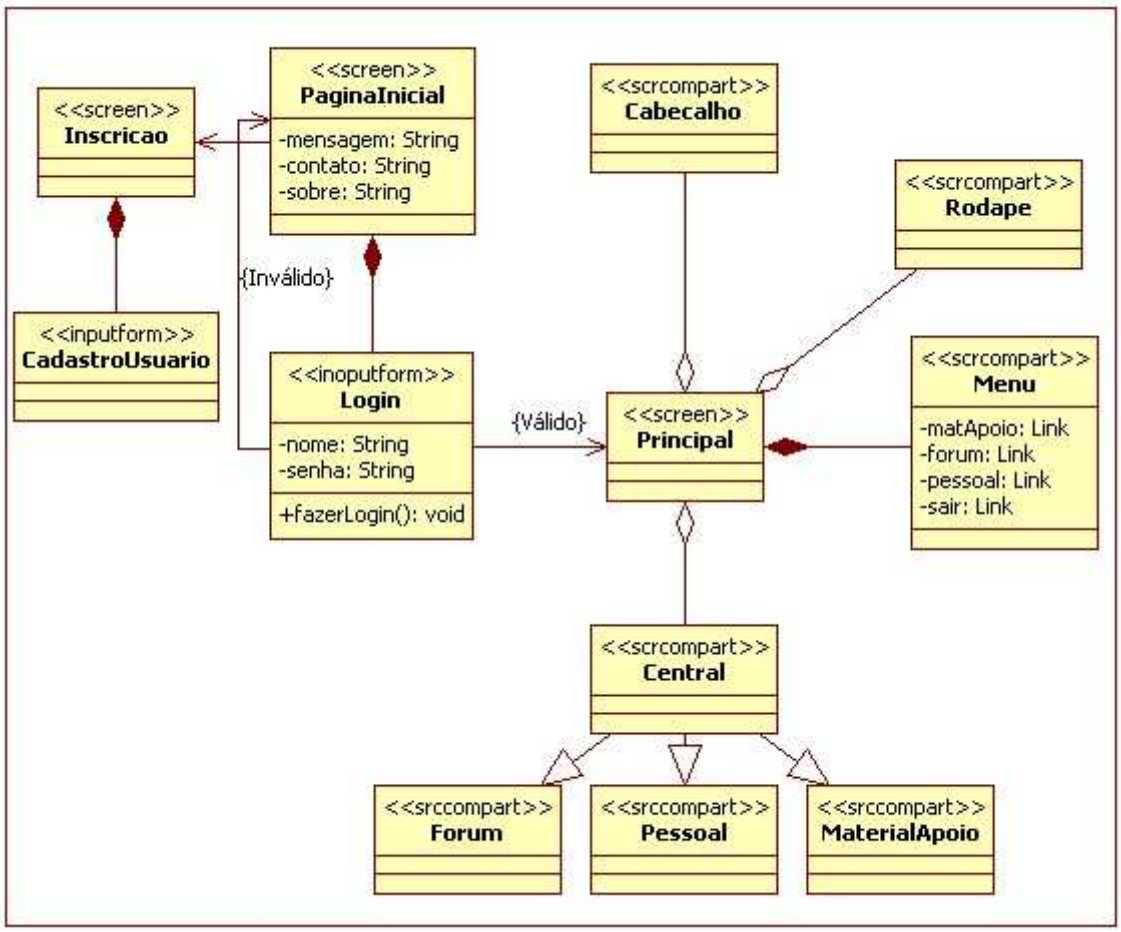

Figura 4.11: Diagrama do modelo UX.

5 CONCLUSÃO

Como apresentado neste trabalho, as aplicações *web* possuem características de uma aplicação de *software* convencional e, portanto, requerem o uso de métodos sólidos e confiáveis para seu desenvolvimento. Porém, essas aplicações também possuem características que as tornam distintas das outras aplicações, tais como requisitos de navegação, usabilidade elaborada, navegação em um ambiente imprevisível, uso de imagens, sons e animações, e exigem que os métodos disponíveis para seu desenvolvimento atendam tais características.

Nesse contexto, o estudo realizado durante o desenvolvimento deste trabalho procurou esclarecer as etapas produtivas envolvidas na criação de uma aplicação *web*. Como mencionado anteriormente, a principal motivação foi a existência de um processo padrão disponível na literatura, que pode tornar o desenvolvimento *web* uma atividade não trivial. Para alcançar esse objetivo, foi necessário o levantamento e análise de um conjunto de métodos, por meio de estudos encontrados na literatura.

A partir dos estudos realizados, pontos importantes foram observados, como a notação utilizada pelos métodos. Métodos mais recentes utilizam como base a linguagem UML, favorecendo o processo de aprendizado, uma vez que essa linguagem é comumente empregada tanto no meio acadêmico quanto no industrial. Há outros métodos que utilizam notação própria e exigem um tempo de aprendizado dos elementos que compõem suas etapas e seus diagramas, dado que materiais de consulta disponíveis para alguns deles são escassos. Ainda em relação à notação, alguns métodos possuem notações próprias que não são facilmente produzidas sem o apoio de uma ferramenta de modelagem, o que torna o uso dessas ferramentas imprescindível para a produção dos diagramas existentes nos métodos.

Outro ponto a ser considerado é que poucos métodos propõem sua utilização concomitante a um processo associado. Dado que já existem processos fundamentados na indústria como o RUP, qualquer método que não sugira o uso de um processo específico, pode fazer uso de processos existentes, de acordo com instruções de um gerente de projetos, equipe de desenvolvimento ou projetista.

Os resultados dos estudos realizados neste trabalho foram sintetizados no arcabouço do estudo apresentado no Capítulo 4. Esse arcabouço foi composto por um conjunto de características, dividido em características de notação e de modelos de engenharia, e nesse conjunto

procurou-se agregar o máximo de características encontradas, percebidas e exigidas para o desenvolvimento de aplicações *web* durante os estudos realizados. A cada característica procurou-se estabelecer uma descrição sucinta.

Ao final deste trabalho e como resultado direto da aplicação do arcabouço, considerando o domínio do *site* flowJava, o método WAE foi considerado como o método com melhores características para desenvolvimento desse tipo de aplicação. Observa-se que esse resultado é dependente da experiência do avaliador.

Além disso, como mencionado anteriormente, o método adotado não referencia a arquitetura das aplicações *web* e, portanto, essa não foi considerada no arcabouço.

6 REFERÊNCIAS

ARAÚJO, A. C. M. *Framework de Análise e Projeto Baseado nio RUP para Desenvolvimento de Aplicações Web.* Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, 2001.

BRAMBILLA, M.; COMAI, S.; FRATERNALI, P. Hypertext semantics for web applications. In: SEBD *Italian National Conference on DataBase Systems*, 2002.

CAGNIN, M. I.; BRAGA, R. T. V.; GERMANO, F.; CHAN, A.; MALDONADO, J. C. Extending Patterns with Testing Implementation. In: *SugarLoafPlop'2005, V Conferencia Latino-Americana em Linguagens de Padrões para Programação*, Campos do Jordão/SP - Brasil, submetido, 2005.

CERI, S.; Fraternali, P.; Bongio, A. *Web* modeling language (webml): a modeling language for designing *web sites*. In: *Proceedings of the 9th international World Wide Web conference on Computer networks: the international journal of computer and telecommunications netowrking,* Amsterdam, The Netherlands, The Netherlands: North-Holland Publishing Co., 2000a, p. 137–157.

CERI, S.; Fraternali, P.; Bongio, A. *Web* Modeling Language (WebML): a Modeling Language for Designing Web Sites. In: *Proceedings of the 9th international World Wide Web Conference,* Elsevier, 2000b.

CONALLEN, J. *Buildind Web applications with UML.* 2nd. ed. Addison-Wesley, 2002.

CUTTER Consortium. *Poor Project Management Number-one Problem of Outsourced Eprojects,* Cutter Research Briefs, November, 2000. Disponível em <http://www.cutter.com/research/2000/crb001107.html>. Acessado em 17/04/2009.

DESHPANDE, Y.; MURUGESAN, S.; GINIGE, A.; HANSEN, S.; SCHWABE, D.; GAEDKE, M.; WHITE, B. Web engineering. *ArXiv Computer Science e-prints*, 2003.

DOMINGUES, A. L. S. *Aplicações Web: Definição e Análise de Recursos de Teste e Validação.* Tese de Doutoramento, ICMC/USP, São Carlos/SP - Brasil, em andamento, 2005.

DOMINGUES, A. L. S.; Bianchini, S. L.; Costa, M. L. S.; Ferrari, F. C.; Maldonado, J. C. *Web application development methods: A comparision. In: Webmedia'2007: Proceedings of the 13o. Brazilian Symposium on Multimedia and the Web,* Gramado, Brazil, 2007.

FONS, J.; Pelechano, V.; Pastor, O.; Albert, M.; Valderas, P. *Extending an OO Method de Develop Web Applications.* In: The Twelfth International World Wide Web Conference, 2003.

FOWLER, M. The new methodology. *Software Development Magazine*, 2000.

GINIGE, A. Engineering a better web site. In: *Asian Pacific Web Conference,* Xian, China, 2000.

GINIGE, A.; MURUGESAN, S. Web engineering: An introduction. In: *IEEE Multimedia*, 2001, p. 14–18.

GINIGE, A. Web engineering: managing the complexity of web systems development. In: *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering,* New York, NY, USA: ACM Press, 2002, p. 721–729.

GRIFFITHS, G.; HEBBRON, B. D.; LOCKYER, M. A.; OATES, B. J. A simple method & tool for web engineering. In: *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, New York, NY, USA: ACM Press, 2002, p. 755–762.

ISAKOWITZ, T.; STOHR E. & BALASUBRAMANIAN P. RMM: *A Methodology for Strutured Hypermidia Design, Communications of ACM,* vol. 38, no. 8, August 1995.

KAPPEL, G.; MICHLMAYR, E.; PRÖLL, B.; REICH, S. & RETSCHITZEGGER, W. *Web Engineering – Old wine in new bottles.* 4th International Conference on Web Engineering - ICWE 2004, pp. 6-12, 2004.

LEITE, Jair Cavalcanti. *Desenvolvimento e design de sistemas web.* Natal, 2002. Disponível em http://www.dimap.ufrn.br/~jair/. Acesso em: 11 de outubro de 2009 .

MCT - Ministério de Ciência e Tecnologia. *Qualidade e Produtividade no Setor de Software.* 2002. Disponível em <www.mct.gov.br/temas/info/dsi/software/ menu_qualidade.htm>. Acessado em 10/04/2009.

MELO, A. *PHP Profissional.* 1 ed. São Paulo: Novatec , 2007.

MURUGESAN, S.; Ginige, A. *Web engineering: Introduction and perspectives, cáp. 1* Idea a Group Publishing, p. 23, 2005.

MURUGESAN, S.; Deshpande, Y. Second icse workshop on web engineering (workshop session). In: *ICSE '00: Proceedings of the 22nd international conference on Software engineering,* New York, NY, USA: ACM Press, 2000, p. 794–795.

NIELSEN, Jacob. *Designing Web Usability: The practice of simplicity.* Indianapolis: New Riders Publishing, 2000.

PAPAZOGLOU, M. P.; GEORGAKOPOULOS, D. *Introduction. Commun.* ACM, v. 46, n. 10, p. 24–28, 2003.

PAULA FILHO, W. P.; Engenharia de Software – Fundamentos, métodos e padrões. 63 ed. LTC, 2003.

PFLEEGER, S. L. *Engenharia de Software: teoria e prática.* Pearson; 2004.

POWELL, T.; JONES, D.; CUTTS, D. *Web site engeneering: beyond web page design.* Prentice Hall, p. 11 – 38, 1998.

PRESSMAN, R. S. *Software Engeneering – A Practioner's Approach.* 6 ed. McGrall-Hill, 2005.

RODRIGUEZ, D.; HARRISON, R. & SATPATHY, M. *A Generic Model and Tool Support for Assessing and Improving Web Processes.* Proceedings of the Eighth IEEE Symposium on Software Metrics. IEEE, 2002.

RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W. *Object-Oriented Modeling and Design.* Englewood Cliffs, New Jersey, USA: Prentice Hall International, 1991.

RUMBAUGH, J.; JACOBSON, I.; BOCH, G. *The unified modeling language reference manual.* Addison-Wesley Object Technology Series, 1998.

SHOHOUD, Yasser. *Real World XML Web Services.* Pearson Education Inc., 2003.

SCHWABE, D.; Rossi, G.; Barbosa, S. D. J. *Systematic hypermedia application design with oohdm.* In: HYPERTEXT '96: Proceedings of the the seventh ACM conference on Hypertext, New York, NY, USA: ACM Press, 1996, p. 116–128.

SCHWABE, D.; Rossi, G. *An Object Oriented Approach to Web-Based Application Design.* Theory and Practice of Object Systems, Special Issue on the Internet 4, p. 34, 1998.

SUN MICROSYSTEMS Interactive Web Application Architectures. online, disponível em http://java.sun.com/webservices/docs/1.2/tutorial/doc/IntroIWA2.html - Último acesso em 20/04/2009.

TANENBAUM, A. S.: *Redes de Computadores*, Editora Campus, 1997.

TORRES, V.; FONS, J.; ASENSI, O.; PELECHANO, V. Getting ready web engineering methods for the semantic web. putting ontologies into practice. In: *International Workshop on Web Oriented Software Technology (IWWOST)*, 2004.

W3C Recommendation. *W3C in 7 points.* 2007. Disponível em: <http://www.w3.org/Consortium/Points/>. Acessado em: 13 abril 2009.

WINMAN, W. E. *Manual de CGI.* Makron Books; 1997.

APENDICE A

Telas do *website* flowjava hospedado em http://www.flowjava.com

Figura A.1: Menu principal - home do *site* flowjava.com.

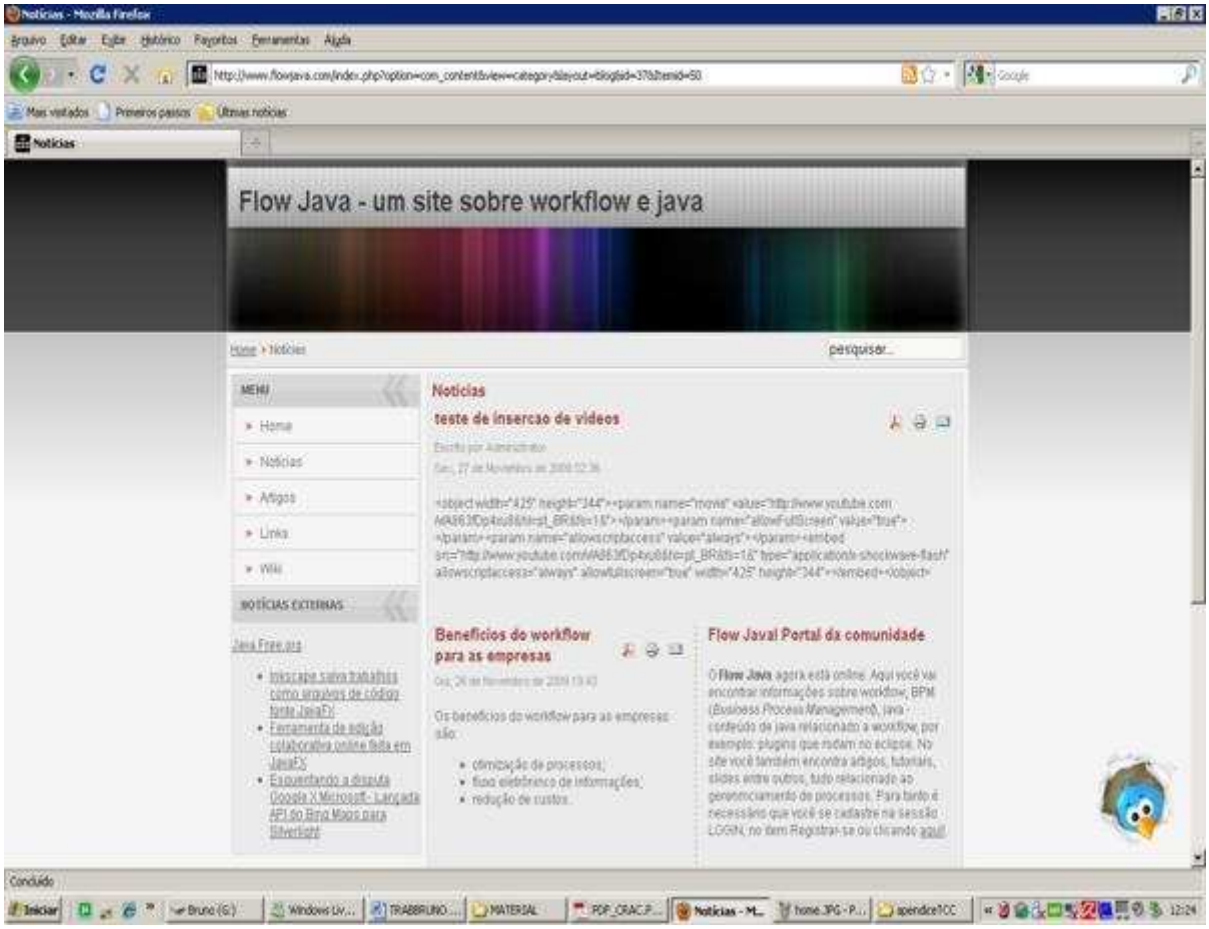

Figura A.2: Menu principal - notícias do *site* flowjava.com.

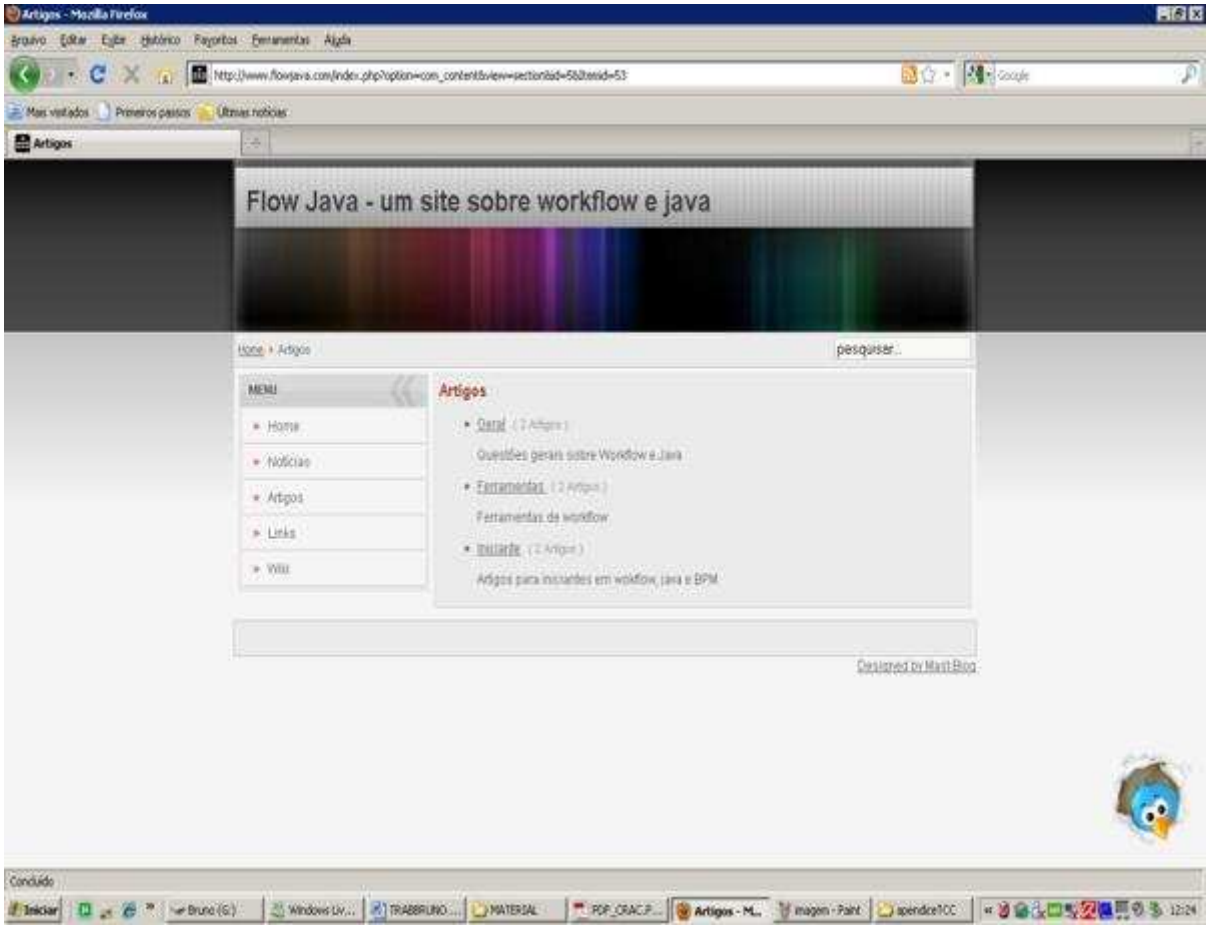

Figura A.3: Menu principal - artigos do *site* flowjava.com.

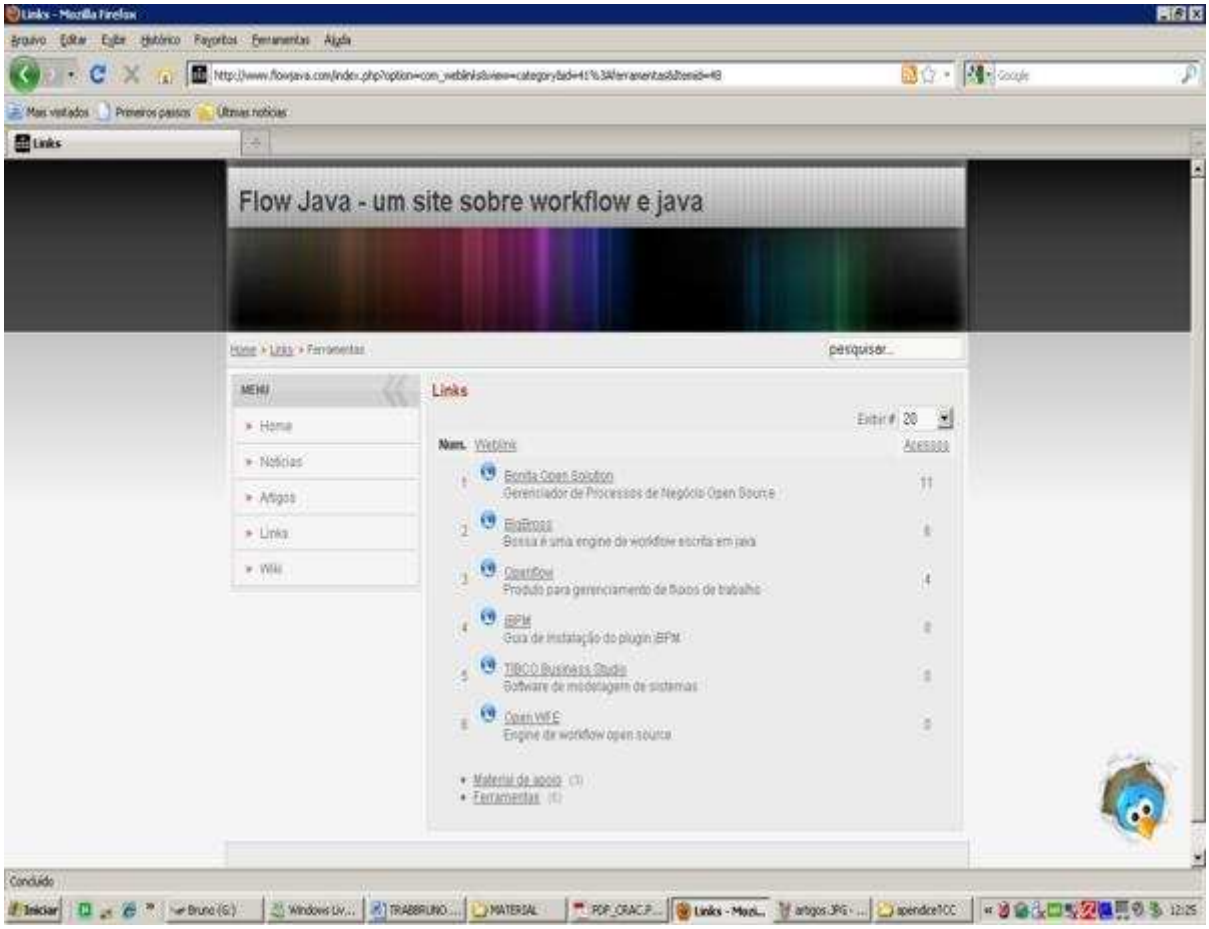

Figura A.4: Menu principal - links do *site* flowjava.com.

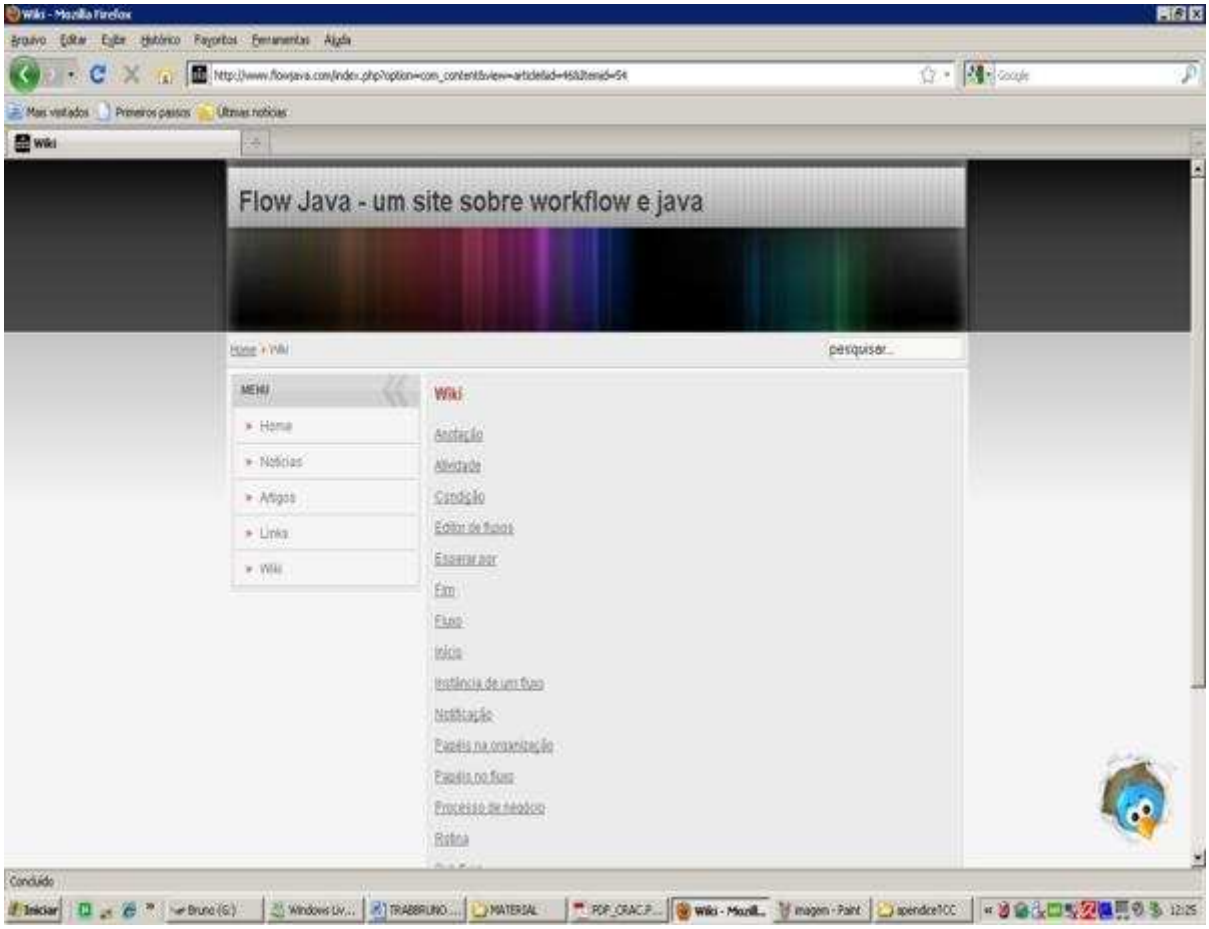

Figura A.5: Menu principal - wiki do *site* flowjava.com.

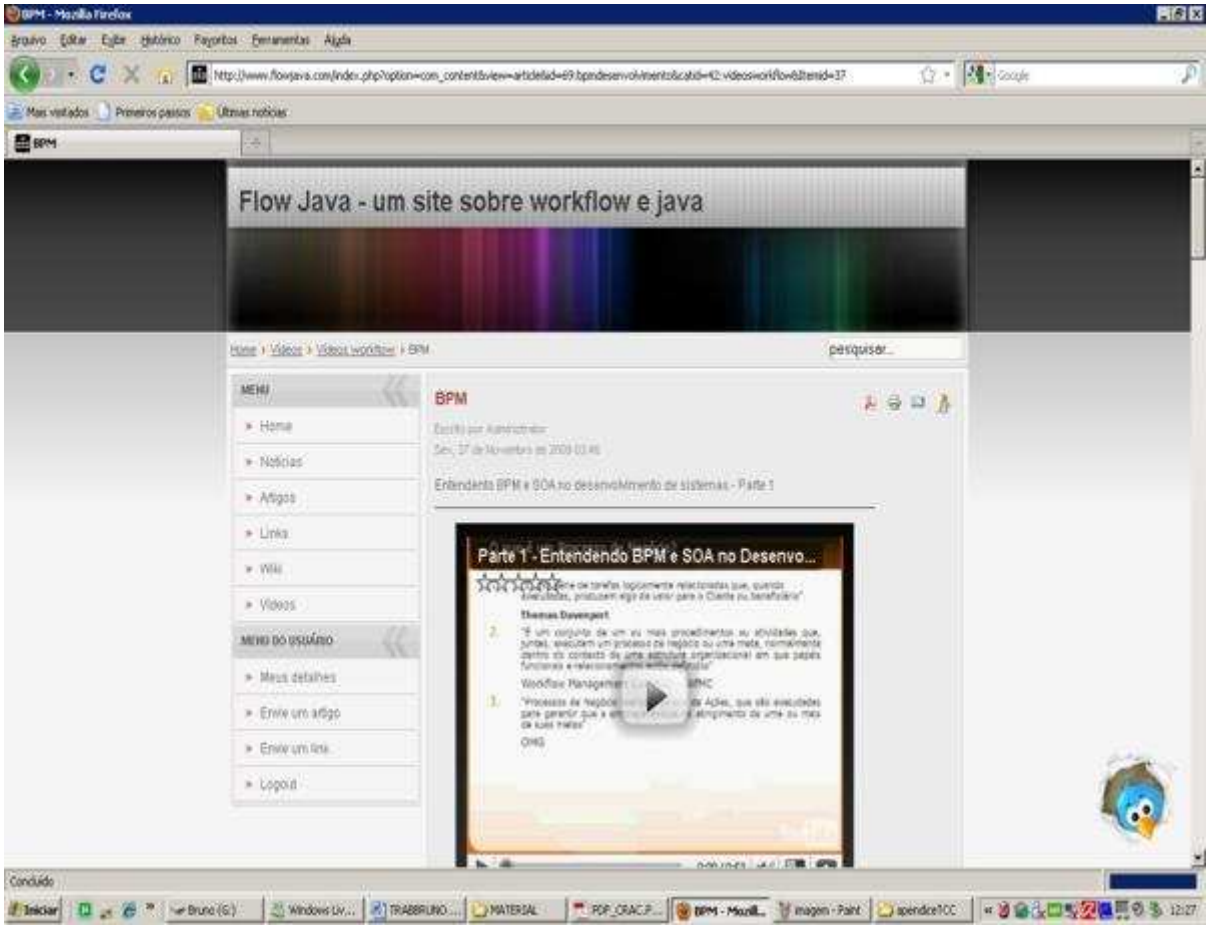

Figura A.6: Menu usuário registrado - vídeos do *site* flowjava.com.

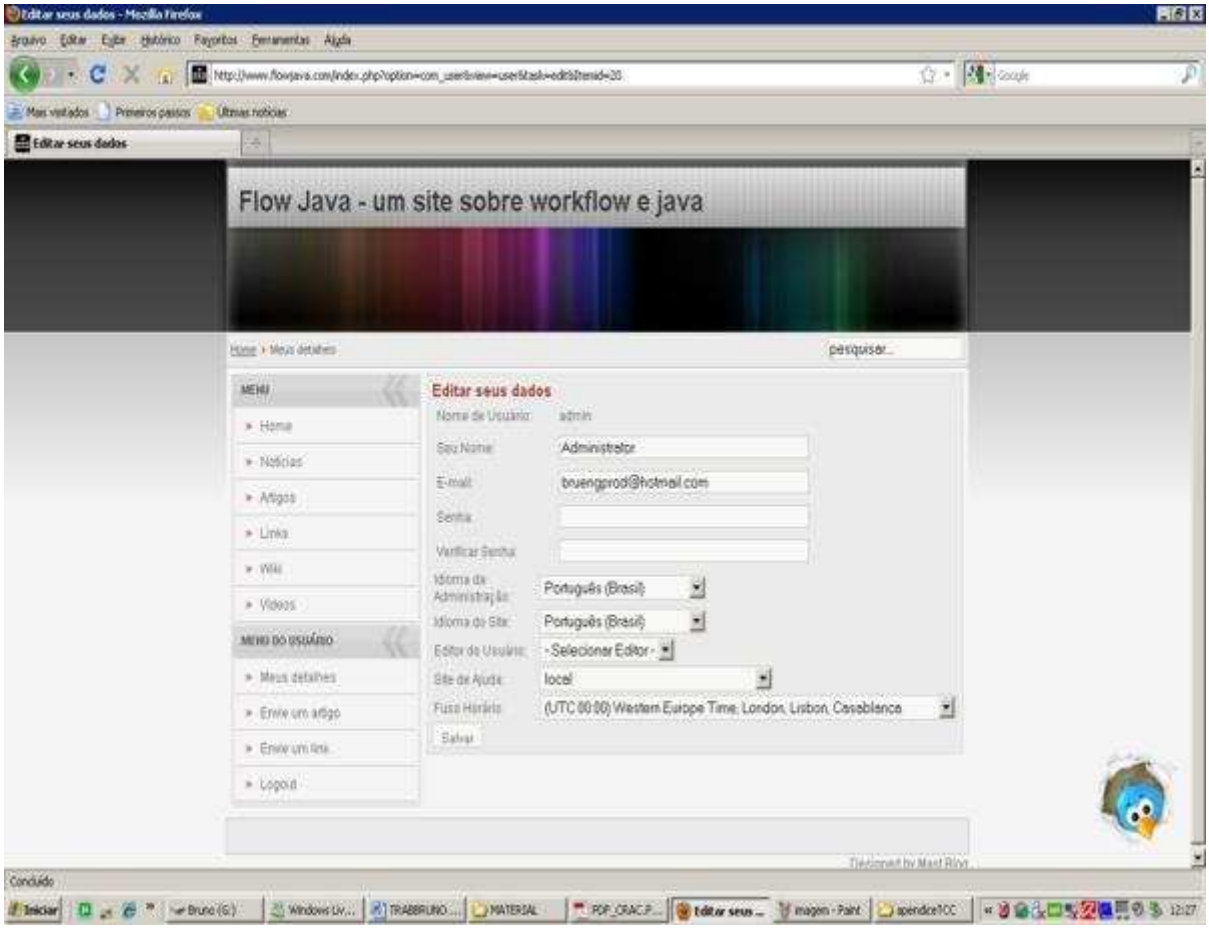

Figura A.7: Menu usuário registrado – meus detalhes do *site* flowjava.com.

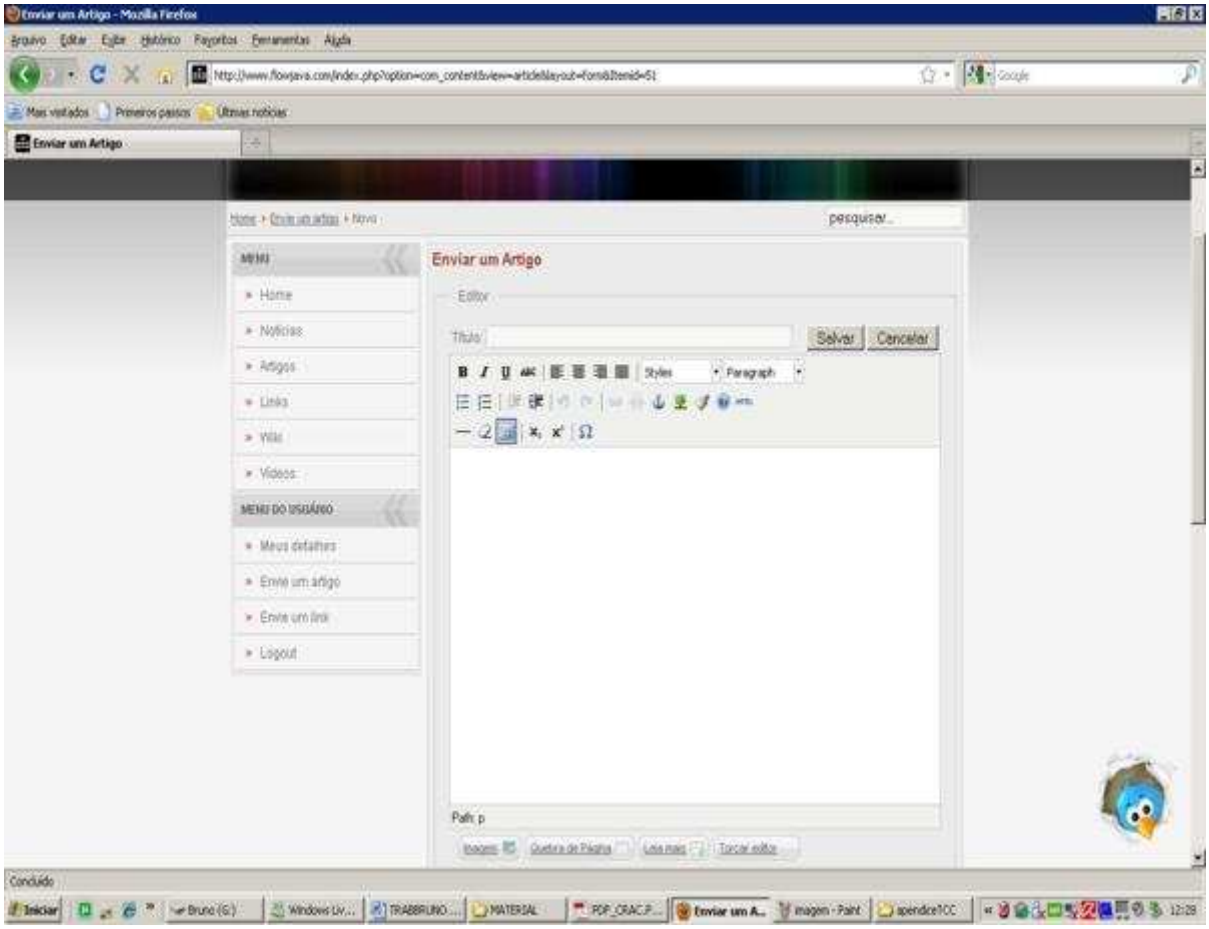

Figura A.8: Menu usuário registrado – envie um artigo do *site* flowjava.com.

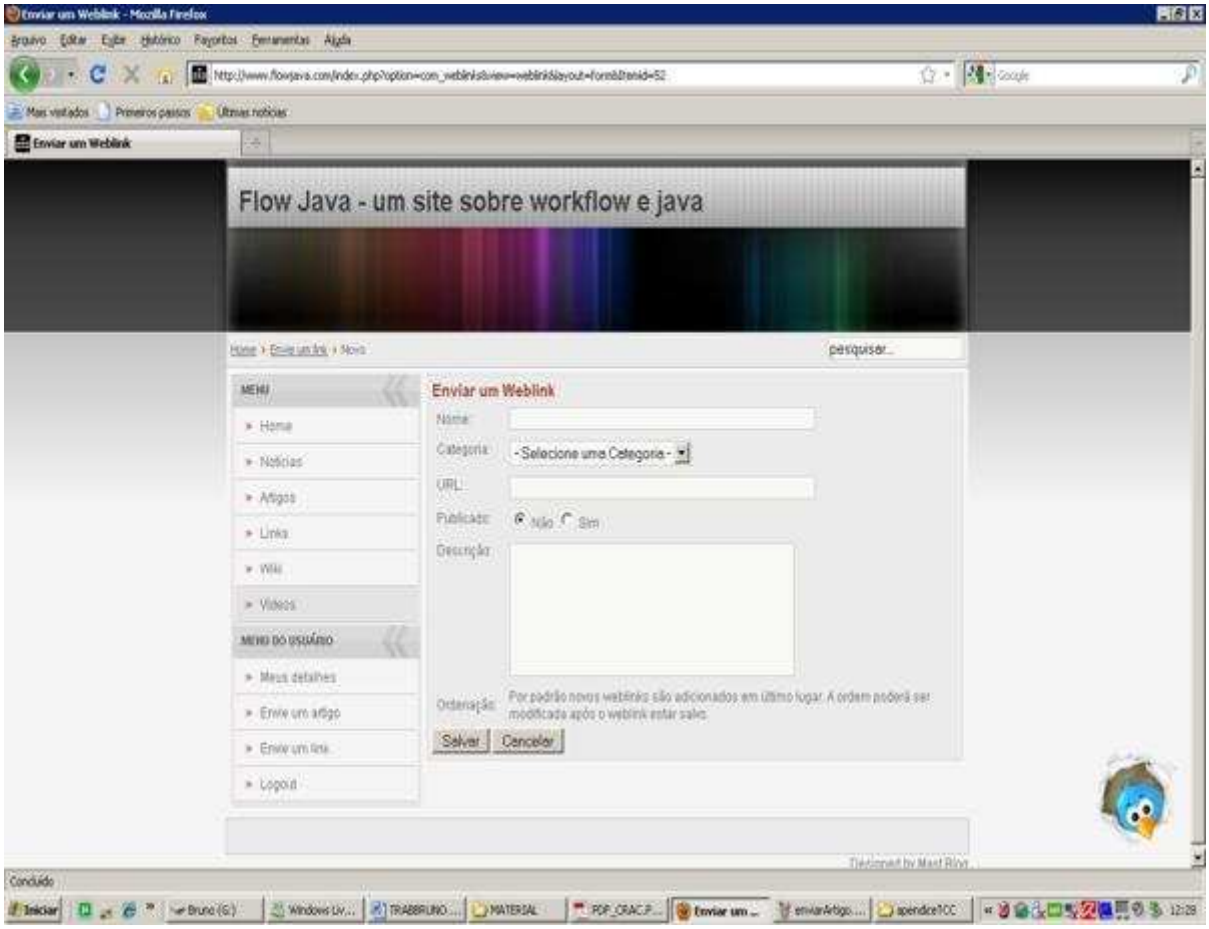

Figura A.9: Menu usuário registrado – envie um link do *site* flowjava.com.